Effiziente Vorkonditionierung von Finite-Elemente-Matrizen unter Verwendung hierarchischer Matrizen

Von der Fakultät für Mathematik und Informatik
der Universität Leipzig
angenommene

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

DOCTOR rerum naturalium
(Dr. rer. nat.)

im Fachgebiet

Mathematik

vorgelegt

von Dipl.-Inf. Thomas Fischer

geboren am 19.04.1977 in Neuhaus am Rennweg

Die Annahme der Dissertation wurde empfohlen von:

1. Prof. Dr. Mario Bebendorf, Universität Bonn
2. Prof. Dr. Dr. h.c. Wolfgang Hackbusch, Universität Leipzig

Die Verleihung des akademischen Grades erfolgt mit Bestehen
der Verteidigung am 15.09.2010 mit dem Gesamtprädikat magna cum laude.

Inhaltsverzeichnis

Einleitung

In vielen Wissenschaften führt die Modellbildung auf partielle Differentialgleichungen. So werden vor allem in der Physik (Schwingungen, Wärmeleitung, Wellenausbreitung, Diffusion, Schrödinger-Gleichung), aber auch in vielen anderen Disziplinen wie der Ökonomie (Black-Scholes-Modell zur Bewertung von Finanzoptionen), der Biologie (Populationsdynamik) und der Medizin (Zellwachstum) die Lösungen partieller Differentialgleichungen benötigt. Durch eine schnelle Lösung von partiellen Differentialgleichungen bietet sich die Möglichkeit der Simulation und Optimierung der modellierten Prozesse am Rechner. Dabei lassen sich optimale Lösungen am Computermodell einfacher finden als in einem Versuch mit Prototypen.

Im Gegensatz zu gewöhnlichen Differentialgleichungen existiert selbst für lineare partielle Differentialgleichungen mit konstanten Koeffizienten keine allgemeine Lösungstheorie. Analytische Lösungen partieller Differentialgleichungen sind nur auf einfachen Gebieten bekannt. Daher muss die Lösung numerisch bestimmt werden.

Nichtlineare partielle Differentialgleichungen können mit Hilfe des Newton-Verfahrens gelöst werden. Dazu muss in jedem Schritt des Newton-Verfahrens die Lösung einer linearen partiellen Differentialgleichung berechnet werden. Deshalb beschränken wir uns auf die Behandlung linearer partieller Differentialgleichungen.

Zur praktischen Berechnung von Lösungen werden partielle Differentialgleichungen diskretisiert. Die dabei entstehenden linearen Gleichungssysteme sind umso größer, je feiner diskretisiert wird. Die linearen Gleichungssysteme besitzen pro Zeile und Spalte nur wenige von Null verschiedene Einträge und sind schlecht konditioniert. Zur Lösung dieser Systeme werden immer häufiger iterative Verfahren in Verbindung mit Vorkonditionierern eingesetzt. Mit der Technik der hierarchischen Matrizen lassen sich effiziente Vorkonditionierer konstruieren, die eine schnelle Konvergenz iterativer Verfahren sichern.

Hierarchische Matrizen basieren auf einer Partition der Matrix in Blöcke und der Niedrigrang-Approximation der dafür zulässigen Blöcke. Dabei ist eine günstige Partition für die Laufzeit der arithmetischen Operationen mit hierarchischen Matrizen besonders wichtig. Bisher wird die Partition der Matrix auf geometrischen Informationen vorgenommen.

In dieser Arbeit wird ein Verfahren zur Partitionierung vorgestellt, das eine Partition ohne geometrische Informationen, d.h. rein algebraisch, konstruiert. In diesem neuen Ansatz werden die Matrixeigenschaften zur Partitionierung ausgenutzt. Dies erleichtert auch die Integration einer Implementierung des Verfahrens in externe Software, weil nur auf eine kleine Schnittstelle zugegriffen werden muss.

Das erste Kapitel enthält eine Einteilung partieller Differentialgleichungen. Im weiteren Verlauf wird die Diskretisierung elliptischer partieller Differentialgleichungen mit der Finiten-Elemente-Methode skizziert. Die Lösung des aus der Diskretisierung resultieren-

den linearen Gleichungssystems bildet den Hauptaspekt dieser Arbeit.

Da viele der in der Arbeit auftretenden algorithmischen Probleme auf Graphen zurückgeführt werden können, behandelt das zweite Kapitel die benötigten graphentheoretischen Begriffe und führt zwei Algorithmen zur Bestimmung von Abständen in Graphen ein. Dies sind die Breitensuche zur Bestimmung von Abständen in ungewichteten Graphen sowie der Algorithmus von Dijkstra zur Bestimmung von Abständen in gewichteten Graphen.

Zur Lösung der aus der Finite-Elemente-Methode resultierenden Gleichungssysteme werden in der Praxis direkte Verfahren, Mehrgitter-Verfahren oder iterative Methoden eingesetzt. Diese werden im Kapitel 3 vorgestellt. Daneben befassen wir uns in diesem Kapitel mit Vorkonditionierern, weil sie die von der Kondition der Matrix abhängige Konvergenz iterativer Verfahren verbessern können. Die Vorkonditionierung dient nicht nur zur Steigerung der Effizienz iterativer Löser, sondern auch zur Verbesserung der Robustheit. Die Technik der hierarchischen Matrizen ist zur Vorkonditionierung besonders geeignet, da sich der Vorkonditionierer in linear-logarithmischer Komplexität erstellen und anwenden lässt.

Im vierten Kapitel werden hierarchische Matrizen vorgestellt. Die Technik der hierarchische Matrizen basiert auf einer Partitionierung der Matrix in Blöcke, die entweder klein sind oder sich zur Niedrigrang-Approximation eignen. Mit Hilfe der hierarchischen Matrizen kann eine approximierte Inverse bzw. eine approximierte LU-Zerlegung in linear-logarithmischer Zeit berechnet werden. Bisherige Ansätze verwenden zur Matrixpartitionierung die Geometrie.

Auf Grund der zunehmenden geometrischen Komplexität der zu modellierenden Bauteile oder adaptiver Verfeinerungen werden verstärkt rein algebraische Verfahren eingesetzt. Die im fünften Kapitel eingeführte algebraische Konstruktion hierarchischer Matrizen folgt dieser aktuellen Entwicklung. Dazu wird zunächst eine algebraische Zulässigkeitsbedingung vorgestellt. Auf deren Basis wird die Existenz der Faktoren L und U der LU-Zerlegung im hierarchischen Format nachgewiesen. Zudem geben wir eine konkrete algorithmische Realisierung der Partition an. Diese nutzt lediglich die Besetzungstruktur der Matrix aus. Die neue Methode ermöglicht somit die Behandlung von Problemen, ohne auf die Kenntnis der Geometrie angewiesen zu sein.

Im sechsten Kapitel werden hierarchische Matrizen, basierend auf geometrischer bzw. algebraischer Matrixpartition, verglichen. Es zeigt sich, dass bei ähnlicher Wirkung des Vorkonditionierers die algebraische Konstruktion sowohl bzgl. der Zeit als auch der Speicherbelegung effizienter ist.

Zudem werden die Laufzeit sowie der Speicherverbrauch der neuen Methode mit denen eines direkten Verfahrens (PARDISO) und denen eines mit Algebraischen Mehrgitter-Verfahren (AMG) vorkonditionierten CG-Verfahrens bzgl. der Asymptotik und der Robustheit verglichen. Hierarchische Vorkonditionierer erweisen sich dabei als die speichereffizienteste Methode. Das AMG-Verfahren ist asymptotisch für ein zu lösendes lineares System das schnellste Verfahren. Sind jedoch mehrere rechte Seiten - in den numerischen Experimenten weniger als zehn - zu lösen, lohnt sich auf Grund der Wiederverwendbarkeit der Einsatz hierarchischer Vorkonditionierer. Zudem ist das CG-Verfahren mit hierarchischer Vorkonditionierung deutlich robuster als das AMG vorkonditionierte

CG-Verfahren.

Die numerischen Experimente zur Parallelisierbarkeit der Konstruktion des hierarchischen Vorkonditionierers zeigen für kleine Prozessoranzahlen eine sehr gute parallele Effizienz. Diese sinkt für 12 Prozessoren auf ca. 50% ab.

Der Vergleich von AMG-vorkonditioniertem und hierarchisch vorkonditioniertem CG-Verfahren innerhalb des Newton-Verfahrens ergibt für die Gesamt-Rechenzeiten der Experimente Vorteile für das hierarchisch vorkonditionierte CG-Verfahren.

1 Partielle Differentialgleichungen

Häufig führt die Modellbildung in Naturwissenschaften auf partielle Differentialgleichungen (PDEs), d.h. auf Gleichungen der Form

$$F(x_1, x_2, \dots, x_d, u, u_{x_1}, u_{x_2}, \dots, u_{x_d}, u_{x_1x_1}, u_{x_1x_2}, \dots, u_{x_1x_d}, \dots) = 0. \tag{1.1}$$

Gesucht wird eine Funktion u, die (1.1) erfüllt. Im Gegensatz zu gewöhnlichen Differentialgleichungen existiert selbst für lineare PDEs mit konstanten Koeffizienten keine allgemeine Lösungstheorie. Analytische Lösungen von (1.1) sind nur auf einfachen Gebieten bekannt und müssen daher numerisch bestimmt werden. Zur Lösung nichtlinearer partieller Differentialgleichungen wird oft das Newton-Verfahren eingesetzt. In jeder Iteration des Newton-Verfahrens ist eine lineare Differentialgleichung zu lösen. Dazu werden die linearen Differentialgleichungen zunächst diskretisiert. Aus der Diskretisierung resultieren sehr große Gleichungssysteme. Die Lösung einer linearen partiellen Differentialgleichung wird also auf die Lösung eines algebraischen Gleichungssystems reduziert.

Es gibt verschiedene Diskretisierungsverfahren, zum Beispiel das Differenzenverfahren oder die Finite-Elemente-Methode. Da die Methode der Finite-Elemente für kompliziertere Geometrien besser geeignet ist, wird diese in diesem Kapitel kurz skizziert.

Parallel zur zunehmenden Rechenleistung und Speichermenge sollen größere Gleichungssysteme gelöst werden. Wünschenswert wäre es, bei verdoppelter Rechenleistung auch ein Gleichungssystem doppelter Größe lösen zu können. Dies setzt aber eine lineare Komplexität des Lösungsalgorithmus voraus.

1.1 Einteilung partieller Differentialgleichungen

Man nennt (1.1) *linear*, wenn die gesuchte Funktion u und deren partielle Ableitungen nur linear auftreten. Eine partielle Differentialgleichung bezeichnet man als *homogen*, wenn kein Summand auftritt, der nicht mit u oder einer partiellen Ableitung von u multipliziert ist. Die Gleichung (1.1) ist von i-ter *Ordnung*, wenn die höchsten auftretenden Ableitungen i-ter Ordnung sind. Partielle Differentialgleichungen können nach ihrer Ordnung oder nach den modellierten Prozessen eingeteilt werden.

Wir betrachten die partielle Differentialgleichung zweiter Ordnung

$$-\sum_{j,k=1}^{d} c_{jk}(\boldsymbol{x}) \frac{\partial^2}{\partial x_j \partial x_k} u(\boldsymbol{x}) + \sum_{i=1}^{d} b_i(\boldsymbol{x}) \frac{\partial}{\partial x_i} u(\boldsymbol{x}) + d(\boldsymbol{x}) u(\boldsymbol{x}) = f(\boldsymbol{x}), \quad \boldsymbol{x} \in \Omega. \tag{1.2}$$

Die zugeordnete $d \times d$ Matrix $C(\boldsymbol{x}) := (c_{jk}(\boldsymbol{x}))$ kann als symmetrisch angenommen werden, da für zweimal stetig differenzierbare Funktionen $u_{x_j x_k} = u_{x_k x_j}$ gilt. Die Eigenwerte von $C(\boldsymbol{x})$ sind reell.

Man bezeichnet (1.2) als *elliptisch*, falls alle Eigenwerte λ_C von C für alle $\boldsymbol{x}$ im Gebiet Ω positiv sind. Verschwindet für alle $\boldsymbol{x} \in \Omega$ ein Eigenwert von $C(\boldsymbol{x})$ und sind die weiteren Eigenwerte von gleichem Vorzeichen, nennt man (1.2) *parabolisch*. Nimmt ein Eigenwert von $C(\boldsymbol{x})$ das entgegengesetzte Vorzeichen der übrigen Eigenwerte für alle $\boldsymbol{x} \in \Omega$ an, heißt (1.2) *hyperbolisch*.

Elliptische partielle Differentialgleichungen beschreiben stationäre Zustände, Gleichungen des parabolischen Typs modellieren Diffusionsprozesse und hyperbolische Gleichungen stammen von Transportproblemen.

Wir bezeichnen

$$\mathcal{L} := -\sum_{j,k=1}^{d} c_{jk}(\boldsymbol{x}) \frac{\partial^2}{\partial x_j \partial x_k}$$

als *Hauptteil* von (1.2) und die Darstellung

$$\mathcal{L} = -\sum_{j,k=1}^{d} \frac{\partial}{\partial x_j} \left(c_{jk}(\boldsymbol{x}) \frac{\partial}{\partial x_k} \right) \tag{1.3}$$

als *Divergenzform*.

Ebenso wie bei gewöhnlichen Differentialgleichungen ist die Angabe von zusätzlichen Bedingungen für die Existenz einer eindeutigen Lösung notwendig. Werden die Werte auf dem Rand vorgegeben, dann spricht man von *Dirichlet-Randbedingungen*

$$u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega. \tag{1.4}$$

Die Vorgabe von Ableitungen auf dem Rand bezeichnet man als *Neumann-Randbedingungen*

$$\frac{\partial}{\partial \boldsymbol{\nu}} u(\boldsymbol{x}) = g(\boldsymbol{x}), \quad \boldsymbol{x} \in \partial\Omega,$$

wobei $\boldsymbol{\nu}$ die Normale auf dem Rand ist. Bei zeitabhängigen Problemen besteht die Möglichkeit, Anfangswerte vorzugeben.

Man bezeichnet das Problem als *sachgemäß gestellt*, falls die eindeutige Lösung stetig von den vorgegebenen Daten abhängt.

1.2 Diskretisierung elliptischer Differentialgleichungen - FEM

Zur Diskretisierung elliptischer Differentialgleichungen stehen die Differenzenmethode [16, Kapitel 1 §3] oder die Finite-Elemente-Methode (FEM) [16, 37] zur Verfügung.

Für das numerische Lösen muss die stetige Abhängigkeit der Lösung von den Randwerten und der rechten Seite vorausgesetzt werden. Diese Voraussetzungen sind Konsequenzen aus dem *Maximumprinzip*.

Satz 1.1 (Maximumprinzip). *Sei $\mathcal{L}$ ein elliptischer Operator der Form* (1.3). *Für $u \in C^2(\Omega) \cap C^0(\partial\Omega)$ sei*

$$\mathcal{L}\, u(\boldsymbol{x}) = f(\boldsymbol{x}) \leq 0, \qquad \boldsymbol{x} \in \Omega.$$

Dann nimmt u auf $\partial\Omega$ sein Maximum an, d.h.

$$u(\boldsymbol{x}) \leq \sup_{\boldsymbol{x}\in\partial\Omega} u(\boldsymbol{x}).$$

Wenn darüber hinaus u sein Maximum an einem inneren Punkt von Ω annimmt und Ω zusammenhängend ist, ist u auf Ω konstant.

Beweis. siehe [16] □

Liegt $\mathcal{L}$ in Divergenzform vor, so ist eine Transformation des Problems (1.2) in die *Variationsformulierung*,

$$\text{finde ein } u \in V \text{ mit } a(u,v) = \langle f|v\rangle\,, \quad \forall v \in V, \tag{1.5}$$

möglich. Typischerweise wählt man für V den über $L_2(\Omega)$ aufgebauten Sobolev-Raum

$$H_0^1(\Omega) = \{u \in L_2(\Omega) :\ u \text{ besitzt eine schwache Ableitung}\}.$$

In $H_0^1(\Omega)$ werden das Skalarprodukt

$$\langle u|v\rangle_1 := \sum_{|\alpha|\leq 1} \langle \partial^\alpha u|\partial^\alpha v\rangle_{L_2}$$

und die Norm

$$\|u\|_1 := \sqrt{\langle u|u\rangle_1}$$

definiert.

Für homogene Dirichlet-Randbedingungen erklärt man die Bilinearform

$$a(u,v) := \int_\Omega \left[\sum_{j,k=1}^{d} a_{jk}\partial_j u\partial_k v + duv\right] \mathrm{d}x \tag{1.6}$$

und die Linearform

$$\langle f|v\rangle := \int_\Omega fv\ \mathrm{d}x. \tag{1.7}$$

Voraussetzung für die Transformation der partiellen Differentialgleichung in die Variationsformulierung ist, dass die Funktion v auf dem Rand verschwindet, quadratisch integrierbar ist und der Gradient existiert. In [37, Kapitel 7] wird die Existenz einer Lösung von (1.5) gezeigt, welche als *schwache Lösung* bezeichnet wird. Zudem ist die Variationsformulierung Grundlage für eine Diskretisierung des Problems.

Beim *Ritz-Galerkin-Verfahren* wird eine Basis $\varphi_1, \ldots, \varphi_n$ eines endlich-dimensionalen Unterraums $S_h \subset V$ gewählt, wobei h ein Diskretisierungsparameter ist. Im Allgemeinen wird gefordert, dass sich nur wenige Träger überschneiden, d.h.

$$c_{\text{sparse}} := \max_{i\in I} |\{j \in I : \operatorname{int}\operatorname{supp}\varphi_i \cap \operatorname{int}\operatorname{supp}\varphi_j \neq \emptyset\}|$$

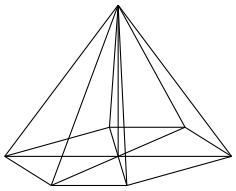


Abb. 1.1: Basisfunktion auf einem Teil der Gebietszerlegung

ist unabhängig von n klein. Ist das Gebiet zwei-dimensional, kann man Dreiecke als Grundelemente verwenden, wie in Abbildung 1.1 skizziert. Bei drei-dimensionalen Gebieten werden beispielsweise Tetraeder als Grundelemente verwendet. Aus Gleichung (1.5) ergeben sich die Forderungen

$$a(u_h, \varphi_k) = \langle f|\varphi_k\rangle\,, \quad k = 1, \ldots, n.$$

Zudem lässt sich die Lösung u_h in S_h als Linearkombination darstellen:

$$u_h = \sum_{i=1}^{n} z_i \varphi_i. \tag{1.8}$$

Wir setzen u_h in die Variationsformulierung ein und erhalten

$$\sum_{i=1}^{n} a(\varphi_i, \varphi_k) z_i = \langle \ell|\varphi_k\rangle \quad k = 1, \ldots, n. \tag{1.9}$$

In Matrix-Vektor-Schreibweise ergibt sich

$$A\boldsymbol{z} = \boldsymbol{b} \tag{1.10}$$

mit $a_{ki} = a(\varphi_i, \varphi_k)$. Für die Matrix A werden die Bezeichnungen *FE-Matrix* (Finite-Elemente-Matrix), *Steifigkeitsmatrix* oder *Systemmatrix* verwendet. Ist die Bilinearform H_0^1-elliptisch, d.h. es gilt $a(v, v) \geq \alpha \|v\|_1^2$, $v \in H_0^1(\Omega)$ mit einer Konstanten $\alpha > 0$, so ist A *symmetrisch* und *positiv definit*, da

$$\begin{aligned}\boldsymbol{z}^T A\boldsymbol{z} = \sum_{i,k=1}^{n} z_i\, a(\varphi_i, \varphi_k) z_k = a\left(\sum_{i=1}^{n} z_i\varphi_i, \sum_{k=1}^{n} z_k\varphi_k\right) = a(u_h, u_h)\\ \geq \alpha\, \|u_h\|_1^2 = \alpha\, \langle u_h|u_h\rangle = \alpha \left\langle \sum_{i=1}^{n} z_i\varphi_i | \sum_{i=1}^{n} z_i\varphi_i \right\rangle = \alpha \sum_{i=1}^{n} z_i^2 \langle \varphi_i|\varphi_i\rangle\,.\end{aligned}$$

Die Qualität der Lösung hängt von der Wahl des Raums S_h ab.

Die in (1.10) definierte $n \times n$-Matrix ist von großer Dimension, besitzt jedoch nur wenige von Null verschiedene Einträge. Die praktische Behandlung solcher Systeme wird erst durch den Verzicht auf die Speicherung von Nulleinträgen und die Definition arithmetischer Operationen, die lediglich die Nicht-Null-Einträge berücksichtigen, möglich. In den

Speicherformaten CRS (compressed row storage) und CCS (compressed column storage) [58] werden nur die Positionen der Nicht-Null-Einträge und die Nicht-Null-Einträge selbst gespeichert.

Wie wir im folgenden Abschnitt sehen werden, setzt man zur Lösung dieser Systeme zunehmend iterative Verfahren ein. Die Konvergenzrate iterativer Verfahren wird durch die Kondition der Matrix bestimmt. Die Kondition Finiter-Elemente-Matrizen steigt mit feiner werdender Diskretisierung. Deshalb müssen diese Gleichungssysteme vorkonditioniert werden. Wir stellen eine effiziente Methode zur Berechnung eines Vorkonditionierers vor, der eine effiziente numerische Behandlung der durch die Diskretisierung elliptischer Probleme zweiter Ordnung entstehenden Gleichungssysteme ermöglicht.

2 Graphen und Graphalgorithmen

2.1 Definitionen

Ziel dieses Kapitels ist die Einführung der benötigten graphentheoretischen Begriffe. Zudem werden die Algorithmen zur *Breitensuche* sowie zur Bestimmung eines kürzesten Weges (*Dijkstras*[1] *Algorithmus*) skizziert.

Wir nennen $G = (V, E)$ einen (endlichen) *Graph*, wobei V die nichtleere (endliche) *Knotenmenge* und $E \subset V \times V$ die (möglicherweise leere) *Kantenmenge* seien. Zwei Knoten $\boldsymbol{v}_i, \boldsymbol{v}_j \in V$ werden *benachbart* (oder *adjazent*) genannt, falls sie durch eine Kante verbunden sind, d.h. $\boldsymbol{e} = (\boldsymbol{v}_i, \boldsymbol{v}_j) \in E$ existiert. Die Menge aller Nachbarn eines Knotens $\boldsymbol{v}$ bezeichnen wir mit $N(\boldsymbol{v})$ oder $\mathrm{Adj}\,[\boldsymbol{v}]$, d.h.

$$N(\boldsymbol{v}) = \mathrm{Adj}\,[\boldsymbol{v}] := \{\boldsymbol{u} \in V : (\boldsymbol{v}, \boldsymbol{u}) \in E\}.$$

Wenn die Knoten $\boldsymbol{v}_i$ und $\boldsymbol{v}_j$ die Endknoten der Kante $\boldsymbol{e}$ sind, dann wird $\boldsymbol{e}$ *inzident* mit $\boldsymbol{v}_i$ bzw. $\boldsymbol{v}_j$ genannt. Zwei Kanten sind *benachbart*, falls die Kanten einen Endknoten gemeinsam haben.

Stimmen die beiden Endknoten einer Kante $\boldsymbol{e}$ überein, so heißt $\boldsymbol{e}$ *Schlinge*. Ein einfacher Graph ist schlingenfrei. Im Folgenden werden nur einfache, ungerichtete Graphen betrachtet. Da in ungerichteten Graphen die Reihenfolge der Knoten $\boldsymbol{v}_i$, $\boldsymbol{v}_j$ im Tupel $\boldsymbol{e} = (\boldsymbol{v}_i, \boldsymbol{v}_j) \in E$ beliebig ist, identifizieren wir eine Kante meist mit der Menge $\{\boldsymbol{v}_i, \boldsymbol{v}_j\}$.

Man nennt $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_k \in E$, mit $\boldsymbol{e}_i = \{\boldsymbol{v}_{i-1}, \boldsymbol{v}_i\}$, $i = 1, 2, \ldots, k-1$, *Kantenfolge* von $\boldsymbol{v}_0$ nach $\boldsymbol{v}_k$ der Länge k. Dabei wird $\boldsymbol{v}_0$ als *Anfangspunkt*, $\boldsymbol{v}_k$ als *Endpunkt* bezeichnet. Sind die Kanten paarweise verschieden heißt $(\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_k)$ *Kantenzug*. Sind im Kantenzug sogar alle Knoten verschieden, d.h.

$$\boldsymbol{e}_i \cap \boldsymbol{e}_j \cap \boldsymbol{e}_\ell = \emptyset,\ \forall i \neq j \neq \ell,$$

so spricht man von einem *Weg* oder *Pfad* und schreibt $p(\boldsymbol{v}_0, \boldsymbol{v}_k)$.

Sei $p(\boldsymbol{v}_0, \boldsymbol{v}_{k-1})$ ein Weg in G, $k > 2$ und $(\boldsymbol{v}_{k-1}, \boldsymbol{v}_0) \in E$. Dann ist der Graph C, bestehend aus den Knoten des Weges und der Vereinigung der Kanten des Weges mit der Kante $(\boldsymbol{v}_{k-1}, \boldsymbol{v}_0)$, ein *Kreis* oder *Zyklus*. Enthält ein Graph keinen Kreis, so nennt man ihn *kreisfrei*.

Man nennt zwei Knoten $\boldsymbol{u}, \boldsymbol{v} \in G$ *zusammenhängend*, falls ein Weg $p(\boldsymbol{u}, \boldsymbol{v})$ existiert. Die so erklärte Beziehung definiert auf der Menge der Knoten eine Äquivalenzrelation. Den von einer Äquivalenzklasse induzierten Teilgraphen bezeichnen wir als *Zusammenhangskomponente* oder *Komponente* von G. Die Anzahl der Komponenten wird durch $\kappa(G)$ gegeben. Der Graph heißt *zusammenhängend*, falls $\kappa(G) = 1$.

[1] Edsger Wybe Dijkstra, 11. Mai 1930 – 06. August 2002, niederländischer Informatiker

Sei $G = (V, E)$ ein zusammenhängender Graph.

(a) Man nennt eine Teilmenge $C \subset E$ *Kantenschnitt*, falls der Graph $G' = (V, E \backslash C\}$ in Komponenten zerfällt.

(b) Die Menge $S \subset V$ bezeichnet man als *Separator* von G, falls $G' = (V \backslash S, E')$ nicht mehr zusammenhängend ist, wobei E' aus E durch das Entfernen aller mit der Knotenmenge S inzidenter Kanten entsteht.

Der Graph $G' = (V', E')$ heißt *Teilgraph* von $G = (V, E)$, falls $V' \subseteq V$ und E' höchstens aus den Kanten $e' \in E$ besteht, deren Endknoten in V' liegen. Der *Abstand* $d : V \times V \to \mathbb{R}_+$ zwischen zwei Knoten $\boldsymbol{u}, \boldsymbol{v}$ ist die Länge des kürzesten Weges $p(\boldsymbol{u}, \boldsymbol{v})$.

$$d(\boldsymbol{u}, \boldsymbol{v}) := \min |p(\boldsymbol{u}, \boldsymbol{v})| \tag{2.1}$$

Lemma 2.1. *Der durch* (2.1) *definierte Abstand ist in einem einfachen, ungerichteten, zusammenhängenden Graphen eine Metrik.*

Beweis. Die Nichtnegativität und Symmetrie folgen unmittelbar aus der Definition. Sei $p_{\min}(\boldsymbol{u}, \boldsymbol{v})$ ein Weg minimaler Länge. Angenommen die Dreiecksungleichung wird durch (2.1) nicht erfüllt. Dann existiert ein $\boldsymbol{w} \in V$, wobei keine Kante des Weges $p_{\min}(\boldsymbol{u}, \boldsymbol{v})$ mit dem Knoten $\boldsymbol{w}$ inzident ist, so dass

$$\underbrace{d(\boldsymbol{u}, \boldsymbol{v})}_{|p_{\min}(\boldsymbol{u},\boldsymbol{v})|} > \underbrace{d(\boldsymbol{u}, \boldsymbol{w})}_{|p_{\min}(\boldsymbol{u},\boldsymbol{w})|} + \underbrace{d(\boldsymbol{w}, \boldsymbol{v})}_{|p_{\min}(\boldsymbol{w},\boldsymbol{v})|} .$$

Folglich ist der Weg von $\boldsymbol{u}$ nach $\boldsymbol{v}$ über $\boldsymbol{w}$ kürzer und $p_{\min}(\boldsymbol{u}, \boldsymbol{v})$ kann nicht der Weg mit minimaler Länge sein. Daher muss die Annahme falsch sein. □

Der *Durchmesser* eines Graphen ist der maximale Abstand zwischen zwei Knoten des Graphen,

$$\operatorname{diam}(G) := \max_{\boldsymbol{v}_i, \boldsymbol{v}_j \in V} d(\boldsymbol{v}_i, \boldsymbol{v}_j). \tag{2.2}$$

Der *Radius* eines Graphen ist der minimale Abstand unter allen maximalen Abständen,

$$r(G) := \min_{\boldsymbol{v}_j \in V} \max_{\boldsymbol{v}_i \in V} d(\boldsymbol{v}_i, \boldsymbol{v}_j). \tag{2.3}$$

Für die in Abschnitt 4.2.1 erklärte Zulässigkeitsbedingung wird der Abstand zwischen zwei Knotenmengen V_1, V_2 benötigt. Dieser ist durch das Minimum der Abstände zwischen Knoten $\boldsymbol{u} \in V_1$ und $\boldsymbol{v} \in V_2$ gegeben, d.h.

$$\operatorname{dist}(V_1, V_2) := \min_{\boldsymbol{u} \in V_1, \boldsymbol{v} \in V_2} d(\boldsymbol{u}, \boldsymbol{v}). \tag{2.4}$$

Der *Knotengrad* $\deg(\boldsymbol{v})$ eines Knotens $\boldsymbol{v}$ ist die Anzahl der mit ihm inzidenten Kanten,

$$\deg(\boldsymbol{v}) := |\{\boldsymbol{e} : \boldsymbol{v} \in \boldsymbol{e}\}| .$$

Als *Paarung* oder *Matching* M von G wird eine Teilmenge der Kanten bezeichnet, wobei $m_i, m_j \in M$ in G paarweise nicht benachbart sind. Kann keine weitere Kante zu M hinzugefügt werden, ohne dass je zwei Kanten benachbart werden, so ist M *maximal*. Gibt es kein Matching mit einer größeren Anzahl an Kanten, so nennen wir M *größtes Matching*. Eine *Knotenüberdeckung* eines Graphen ist eine Teilmenge der Knoten, wobei jede Kante des Graphen mit einem Knoten der Teilmenge inzidiert, d.h.

$$U \subset V \text{ Knotenüberdeckung } \Leftrightarrow \forall e \in E\ \exists u \in U : u \in e.$$

Eng verbunden mit Paarungen ist die *minimale Knotenüberdeckung*, bei der die Menge U minimal wird.

Ein Graph heißt *bipartit*, falls eine Partition $V = A \cup B$ existiert, so dass Knoten aus A nur mit Knoten aus B benachbart sind. In bipartiten Graphen kann ein maximales Matching mit technisch geringerem Aufwand als in allgemeinen Graphen berechnet werden, siehe [45].

Das Tupel $(G, \mathfrak{w})$ wird als *(Kanten-)gewichteter* Graph bezeichnet, wobei $\mathfrak{w} : E \to \mathbb{R}_+$ eine Gewichtsfunktion ist. Wird jeder Kante das Gewicht Eins zugeteilt, erhalten wir den oben definierten Graph.

Die Länge eines Weges $p_{\mathfrak{w}}(\boldsymbol{v}_0, \boldsymbol{v}_k)$ ist dann

$$|p_{\mathfrak{w}}(\boldsymbol{v}_0, \boldsymbol{v}_k)| = \sum_{\ell=1}^{k} \mathfrak{w}\left((\boldsymbol{v}_{\ell-1}, \boldsymbol{v}_\ell)\right).$$

Für Berechnungen von Abständen, Paarungen usw. ist eine Repräsentation eines Graphen im Rechner nötig. Ein Graph kann durch eine Inzidenzmatrix, eine Adjazenzmatrix oder durch Adjazenzlisten dargestellt werden. Zur Implementierung der Algorithmen nutzen wir die Darstellung als Adjazenzmatrix im CRS-Format.

Die $|V| \times |V|$-Matrix $A(G) = (a_{ij})$ mit Einträgen

$$a_{ij} = \begin{cases} \mathfrak{w}(\boldsymbol{e}), & \text{falls } \boldsymbol{e} = (\boldsymbol{v}_i, \boldsymbol{v}_j) \in E, \\ 0, & \text{sonst,} \end{cases}$$

bezeichnet man als *Adjazenzmatrix* des Graphen G.

Beispiel 2.2. Die zum Graphen aus Abb. 2.1 gehörige Adjanzenzmatrix (im vollen Matrixformat) ist

$$\begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

Lemma 2.3. *Die Adjazenzmatrix eines ungerichteten Graphen ist symmetrisch.*

Beweis. Seien $A(G)$ die Adjazenzmatrix des Graphen G und $a_{ij} \neq 0$. Dies ist genau dann der Fall, wenn im Graphen eine Kante existiert, die den Knoten $\boldsymbol{v}_i$ mit dem Knoten $\boldsymbol{v}_j$ verbindet. In einem ungerichteten Graphen ist dies äquivalent zur Aussage: $\boldsymbol{v}_j$ ist mit $\boldsymbol{v}_i$ verbunden, d.h. $a_{ji} \neq 0$. □

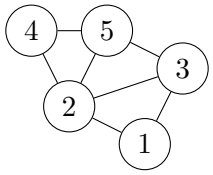


Abb. 2.1: Beispielgraph

Seien I eine Indexmenge und A eine $I \times I$-Matrix. Dann ist

$$G_A := (I, \{(i,j) \in I \times I : i \neq j \text{ und } a_{ij} \neq 0 \text{ oder } a_{ji} \neq 0\}) \tag{2.5}$$

der zu A gehörige (symmetrisierte) *Matrixgraph*.

2.2 Bäume

Ein *Baum* T ist ein kreisfreier und zusammenhängender Graph. Ist ein Teilgraph von T wieder ein Baum, wird er als Teilbaum bezeichnet. Ist ein Knoten $\boldsymbol{r} \in V(T)$ als *Wurzel* markiert, so heißt T auch *Wurzelbaum*. Im Folgenden betrachten wir, falls nichts anderes gesagt wird, ausschließlich Wurzelbäume.

Ein Knoten $\boldsymbol{v}$ befindet sich in der Tiefe ℓ, wenn $\text{lev}\,(\ell) := d(\boldsymbol{r}, \boldsymbol{v})$ gleich ℓ ist. Die Knoten der Tiefe ℓ bilden die ℓ-te Ebene oder Stufe $T^{(\ell)}$, d.h.

$$T^{(\ell)} = \{\boldsymbol{v} \in T : \text{lev}\,(\boldsymbol{v}) = \ell\}. \tag{2.6}$$

Den größten Abstand zwischen Wurzel und einem Knoten $\boldsymbol{v}$ bezeichnet man als *Höhe* oder *Tiefe* des Baums, d.h.

$$L(T) = \max_{\boldsymbol{v} \in T} d(\boldsymbol{r}, \boldsymbol{v}).$$

Die mit $\boldsymbol{v} \in T^{(\ell)}$ benachbarten Knoten $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k \in T^{(\ell+1)}$ heißen *Söhne* und $\boldsymbol{v}$ ist *Vater* der Knoten $\boldsymbol{v}_1, \cdots, \boldsymbol{v}_k$. Die Knoten ohne Söhne heißen *Blätter*,

$$\mathcal{L}(T) := \{\boldsymbol{v} : \boldsymbol{v} \text{ hat keine Söhne}\}.$$

Typische Operationen auf Bäumen sind das Einfügen neuer Knoten, das Entfernen von Knoten, die Suche nach bestimmten Knoten sowie die Traversierung.

Bäume dienen u.a. als temporäre Datenstrukturen zur effizienten Verwaltung von Statusinformationen in Algorithmen. Beispielsweise kann die Laufzeit des Algorithmus von Dijkstra durch die Verwendung partiell geordneter Bäume, sogenannter *Heaps* [18], im Vergleich zur Implementierung mit Listen wesentlich beschleunigt werden.

Im Abschnitt 4.2 werden Wurzelbäume zur Verwaltung von Partitionen einer Indexmenge verwendet.

2.3 Abstandsbestimmung in ungewichteten Graphen: Breitensuche

2.3.1 Algorithmus und Analyse

Ausgehend von einem Startknoten $\boldsymbol{s}$ bestimmt die Breitensuche (breadth first search, BFS) in einem ungerichteten Graphen G einen kürzesten Weg wahlweise zu einem bestimmten Zielknoten $\boldsymbol{t}$ oder zu allen anderen Knoten des Graphen. Existiert zu allen Knoten des Graphen ein Weg, so ist G zusammenhängend. Sei ℓ der maximale Abstand zwischen $\boldsymbol{s}$ zu einem Knoten des Graphen. Es gilt

$$\frac{1}{2}\operatorname{diam}(G) \leq r(G) \leq \ell \leq \operatorname{diam}(G). \tag{2.7}$$

Daher wird ℓ als Näherung für den Durchmesser von G genutzt.

Die erste Phase des Algorithmus umfasst die Initialisierung des kürzesten Wege-Baums π, des Feldes d sowie der Mengen $\mathcal{N}^a$ und $\mathcal{N}^n$. In d werden für alle $\boldsymbol{v} \in V$ die Abstände zu $\boldsymbol{s}$ abgelegt. Der Startknoten wird in die Menge der aktiven Knoten $\mathcal{N}^a$ eingefügt. Die Menge $\mathcal{N}^n$ ist zu Beginn leer und nimmt in der iterativen Phase des Algorithmus die in der nächsten Iteration aktiv werdenden Knoten auf. In V befinden sich die unbesuchten Knoten.

Während der iterativen Phase werden die Knoten des Graphen schichtweise besucht, siehe Abb. 2.2a bis 2.2d. Die im k-ten Schleifendurchlauf besuchten Knoten haben den Abstand k zum Startknoten.

Ist $V \neq \emptyset$, so werden für jeden aktiven Knoten $\boldsymbol{u} \in \mathcal{N}^a$ seine bisher unbesuchten Nachbarn $\boldsymbol{v}$ in den kürzesten Wege-Baum eingefügt sowie der entsprechende Abstand gesetzt. Der Knoten $\boldsymbol{v}$ gilt nun als besucht und wird in der nächsten Iteration aktiver Knoten ($\mathcal{N}^n := \mathcal{N}^n \cup \{\boldsymbol{v}\}$). Der Algorithmus terminiert, falls eine der beiden Mengen $\mathcal{N}^a$ oder V leer ist. Gilt $V = \emptyset$, so ist der Graph zusammenhängend.

Algorithmus 1 BFS (G, s, π)

```
1:  for all v ∈ V do
2:      π[v] := NULL, d[v] := ∞
3:  π[s] := s, d[s] := 0, k := 0
4:  V := V − s, N^a := s, N^n := ∅
5:  while V ≠ ∅ und N^a ≠ ∅ do
6:      k := k + 1
7:      for all u ∈ N^a do
8:          for all v ∈ Adj[u] und v ∈ V do
9:              π[v] := u, d[v] := k
10:             V := V − v, N^n := N^n ∪ v
11:     N^a := N^n, N^n := ∅
```

Die Felder π und d benötigen $\mathcal{O}(|V|)$ Speichereinheiten. Die Verwaltung der beiden Mengen $\mathcal{N}^a$, $\mathcal{N}^n$ nimmt höchstens $\mathcal{O}(|V|)$ Speichereinheiten in Anspruch. Insgesamt ist der Speicherverbrauch linear in der Anzahl der Knoten.

Zu Beginn der Breitensuche sind alle Knoten bis auf den Startknoten unbesucht. Während der iterativen Phase (Zeile 5 bis Zeile 11) werden einmal besuchte Knoten aus der Knotenmenge V entfernt, d.h. es werden $\mathcal{O}(|V|)$ Operationen benötigt. Im ungünstigsten Fall müssen alle von $\boldsymbol{u}$ ausgehenden Kanten überprüft werden, was einem Aufwand von $\mathcal{O}(|E|)$ Operationen entspricht. Die Anzahl der Operationen zur Breitensuche ist daher $\mathcal{O}(|V| + |E|)$.

Für die später betrachteten Matrixgraphen schwach besetzter Matrizen gilt $|E| = \mathcal{O}(|V|)$. Sowohl der Speicherverbrauch als auch die Laufzeit der Breitensuche sind für Matrixgraphen schwach besetzter Matrizen linear in der Anzahl der Knoten.

2.3.2 Beispiel

Im Graphen der Abb. 2.2a ist der Startknoten $\boldsymbol{o}$ markiert. Im ersten Schleifendurchlauf ist $\mathcal{N}^a = \{\boldsymbol{o}\}$, und es werden alle Nachbarn von $\boldsymbol{o}$ in die Menge $\mathcal{N}^n$ aufgenommen, d.h. in Zeile 11 des Alg. 1 ist $\mathcal{N}^n = \{\boldsymbol{c}, \boldsymbol{e}, \boldsymbol{k}, \boldsymbol{n}, \boldsymbol{p}\}$, siehe Abb. 2.2b. In der zweiten Iteration besteht $\mathcal{N}^a$ aus der im vorhergehenden Schritt berechneten Menge $\mathcal{N}^n$. Alle unbesuchten Nachbarn dieser Knoten werden im Schritt zwei besucht und sind in Abb. 2.2c durch grauen Hintergrund ohne schwarzen Rand gekennzeichnet. Im dritten Schleifendurchlauf ist $\mathcal{N}^a = \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{d}, \boldsymbol{f}, \boldsymbol{i}, \boldsymbol{j}, \boldsymbol{\ell}, \boldsymbol{t}, \boldsymbol{w}\}$, und alle noch nicht besuchten Knoten sind Nachbarn eines Knotens aus $\mathcal{N}^a$, siehe Abb. 2.2d. Die schrittweise Konstruktion eines kürzesten Wege-Baums wird durch die in den Abb. 2.2a bis 2.2d schwarz gezeichneten Kanten verdeutlicht.

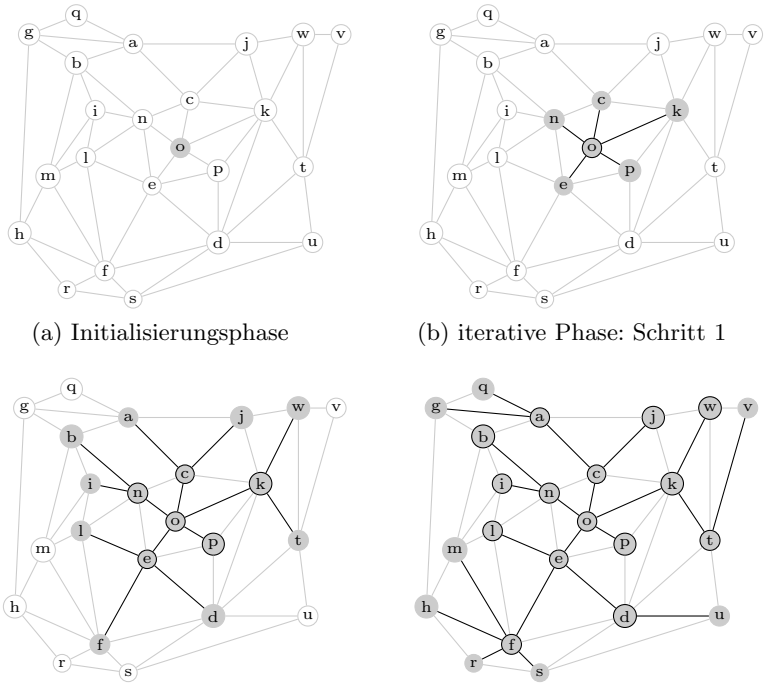
2.4 Abstandsbestimmung in gewichteten Graphen: Algorithmus von Dijkstra

2.4.1 Algorithmus und Analyse

Der Algorithmus von Dijkstra dient der Berechnung von Wegen bzw. Abständen in gewichteten Graphen. Während der Ausführung des Algorithmus wird ein Kürzester-Wege-Baum konstruiert. An diesem sind wir hier nicht interessiert und lassen entsprechende Anweisungen im Algorithmus aus.

Gegeben sei ein Startknoten s und ein Feld d, in dem die Werte für den Abstand gespeichert werden. Zu Beginn des Algorithmus wird das Feld d initialisiert (Zeilen 1 bis 3 des Alg. 2). In der Datenstruktur Q werden die bisher noch nicht markierten Knoten verwaltet. Sie wird in Zeile 4 initialisert. In der iterativen Phase wird pro Schleifendurchlauf der unmarkierte Knoten $\boldsymbol{u}$ mit minimalem Abstand aus Q entfernt (Zeile 6) und $\boldsymbol{u}$ gilt somit als markiert. Sind die Wege von $\boldsymbol{s}$ zu den Nachbarknoten $\boldsymbol{v}$ über $\boldsymbol{u}$ kürzer als ihr bisher berechneter Abstand, werden die Abstandswerte in Zeile 9 angepasst.

Der Algorithmus terminiert, sobald alle Elemente aus der Menge Q entfernt sind. Enthält Q noch Elemente, aber $\boldsymbol{u}$ besitzt keine Nachbarn, so bricht der Algorithmus ab. In diesem Fall ist der Graph nicht zusammenhängend.



(a) Initialisierungsphase

(b) iterative Phase: Schritt 1

(c) iterative Phase: Schritt 2

(d) iterative Phase: Schritt 3

Abb. 2.2: Breitensuche in einem Graphen

Algorithmus 2 DijkstrasAlgorithmus (G, s)

```
1: for all v ∈ V do
2:     d[v] := ∞
3: d[s] := 0
4: Q ← V
5: while Q ≠ ∅ do
6:     u := ExtractMin(Q)
7:     for all v ∈ Adj [u] do
8:         if d[v] > d[u] + w(u, v) then
9:             DecreaseKey(v, d[u] + w(u, v))
```

In Zeile 4 werden $|V|$ Elemente in Q eingefügt, d.h. wir benötigen $\mathcal{O}\left(|V| \cdot \mathcal{W}_{\text{insert}}\right)$ Operationen, wobei $\mathcal{W}_{\text{insert}}$ den Aufwand für das Einfügen eines Knotens in die Datenstruktur Q bezeichnet. Während eines Schleifendurchlaufs (Zeile 5 bis 9) wird genau ein Knoten entfernt. Daher wird ExtractMin $|V|$-mal aufgerufen. Die Anzahl der pro Aufruf von ExtractMin benötigten Operationen bezeichnen wir mit $\mathcal{W}_{\text{ExtractMin}}$. Zudem wird für jede der $|E|$ Kanten die Operation DecreaseKey höchstens einmal aufgerufen, was pro Ausführung von DecreaseKey mit $\mathcal{W}_{\text{DecreaseKey}}$ Operationen verbunden ist.

Der gesamte Aufwand des Algorithmus von Dijkstra ist

$$\mathcal{W}_{\text{Dijkstra}} = \mathcal{O}\left(|V|\left(\mathcal{W}_{\text{insert}} + \mathcal{W}_{\text{ExtractMin}}\right) + |E|\mathcal{W}_{\text{DecreaseKey}}\right).$$

In Listen sind sowohl das Einfügen eines Element als auch die Operation DecreaseKey in $\mathcal{O}(1)$ möglich, aber ExtractMin benötigt lineare Zeit in $|V|$. Die Laufzeit des Algorithmus von Dijkstra ist $\mathcal{O}\left(|V|^2 + |E|\right)$.

Es existieren Datenstrukturen, die ExtractMin in weniger als $\mathcal{O}(|V|)$ Operationen ausführen. Der Algorithmus von Dijkstra benötigt bei Nutzung sogenannter Heaps [18] nur $\mathcal{O}(|V| \log |V| + |E| \log |V|)$ Operationen.

2.4.2 Beispiel

Die Knoten des Graphen aus Abb. 2.3 sollen 8 Städte[2] symbolisieren. Die Gewichte der Kanten zwischen den Knoten geben die Länge der Orthodrome in Kilometern an. Wir wollen einen kürzesten Weg zwischen Hamburg und Dortmund bestimmen.

Zu Beginn der iterativen Phase ist der Startknoten Hamburg der Knoten mit minimalem Abstand. Während des ersten Schleifendurchlaufs werden die Abstände zu Berlin, Bremen und Hannover berechnet. In Abb. 2.3a ist Hamburg grau markiert und die aktuellen Abstandswerte sind neben den Knoten notiert.

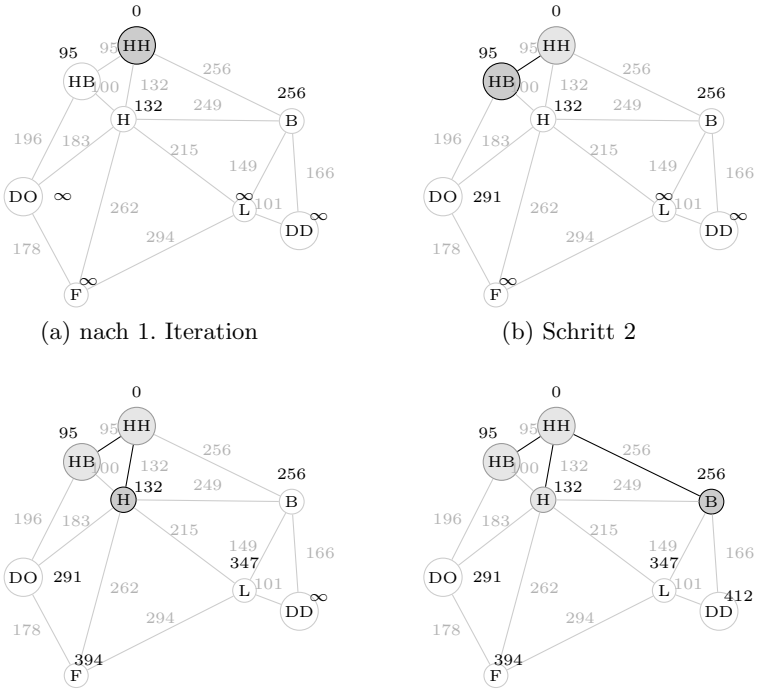
Im zweiten Schritt wird ausgehend vom aktuellen Minimum (Bremen) die Distanz zu Dortmund modifiziert. Da der direkte Weg von Hamburg nach Hannover kürzer ist als der Weg über Bremen nach Hannover, darf der Abstand von Hannover nicht geändert werden, siehe Abb. 2.3b.

Zu Beginn der dritten Iteration liefert extractMin Hannover als Knoten mit dem geringsten Abstand zu Hamburg. Da der direkte Weg zwischen Hamburg und Berlin kürzer ist als der Weg über Hannover muss die Distanz für Berlin nicht angepasst werden. Der Weg von Hamburg nach Dortmund über Bremen ist kürzer als der über Hannover und der Abstand für Dortmund muss ebenfalls nicht aktualisiert werden. Deshalb werden die Distanzen nur für die Städte Frankfurt und Leipzig aktualisiert, siehe Abb. 2.3c.

Im vierten Schritt (Abb. 2.3d) sind vom aktuellen Minimum (Berlin) Leipzig und Dresden erreichbar. Der Weg von Hamburg nach Leipzig über Hannover ist kürzer als über Berlin und deshalb wird die Entfernung von Leipzig nicht modifiziert. Dresdens Entfernung von Hamburg berechnet sich aus der Entfernung von Hamburg zu Berlin und der Distanz zwischen Berlin und Dresden.

In der fünften Iteration liefert extractMin Dortmund als neues Minimum. Da der Wert des Abstands zwischen Hamburg und Dortmund nun nicht mehr kleiner werden kann, wird der Algorithmus beendet.

[2] Legende: B Berlin; HH Hamburg; F Frankfurt; DO Dortmund; HB Bremen; H Hannover; L Leipzig; DD Dresden



(a) nach 1. Iteration

(b) Schritt 2

(c) Schritt 3

(d) Schritt 4

Abb. 2.3: Algorithmus von Dijkstra

3 Überblick über Löser für schwach besetzte Gleichungssysteme

Sei A eine $m \times n$-Matrix. Wir bezeichnen die Menge

$$\operatorname{range}(A) := \{\boldsymbol{y} : \boldsymbol{y} = A\boldsymbol{x}\}$$

als *Bild* von A. Der *Rang* von A ist definiert als die Dimension des Bildes.

$$r := \operatorname{rank} A = \dim \operatorname{range}(A)$$

Es gilt

$$\operatorname{rank} A \leq \min\{m, n\}.$$

Eine $n \times n$-Matrix A heißt *schwach besetzt*, falls sie eine von n unabhängig beschränkte Anzahl von Nicht-Null-Einträgen pro Zeile bzw. Spalte besitzt. Wie wir anhand von Modellproblemen (3.1) sehen, wird diese Eigenschaft bei den Lösungsmethoden berücksichtigt. Die Effizienz der im Abschnitt 3.2 vorgestellten direkten Verfahren kann durch die Umordnung der Zeilen und Spalten gesteigert werden. Die iterativen Verfahren der Abschnitte 3.3 und 3.5 nutzen die effiziente Implementierung der Matrix-Vektor-Multiplikation. Um die effiziente Lösung sehr großdimensionaler Gleichungssysteme zu ermöglichen, wird das Problem mit Mehrgitter-Verfahren (Abschnitt 3.4) hierarchisch gelöst. Auf jeder Hierarchieebene werden dabei stationäre iterative Verfahren eingesetzt.

Sei $\tilde{\boldsymbol{x}}$ eine Näherung für die Lösung $\boldsymbol{x}$ des Gleichungssystems

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{3.1}$$

wobei wir A als regulär voraussetzen. Wir bezeichnen im Folgenden mit $\boldsymbol{e} = \boldsymbol{x} - \tilde{\boldsymbol{x}}$ als Fehler bzw. Fehlervektor und $\boldsymbol{r} = \boldsymbol{b} - A\tilde{\boldsymbol{x}} = A\boldsymbol{e}$ als Residuum. Neben der *Spektralnorm*

$$\|\boldsymbol{v}\|_2 = \sqrt{\langle \boldsymbol{v} | \boldsymbol{v} \rangle}$$

wird auch die *Energienorm*

$$\|\boldsymbol{v}\|_A = \sqrt{\langle A\boldsymbol{v} | \boldsymbol{v} \rangle}$$

verwendet. Hierbei ist $\langle \cdot | \cdot \rangle$ das euklidische Skalarprodukt.

3.1 Modellprobleme

Die von der Kondition der Matrix abhängige Konvergenz iterativer Verfahren lässt sich für die ein-dimensionale Poissongleichung leicht analysieren, da die Kondition des Poisson-Problems analytisch bestimmt werden kann.

Gesucht sei eine Funktion $u : [0,1] \to \mathbb{R}$, die die Differentialgleichung

$$-u''(x) = f(x) \quad \text{für alle } x \in [0,1] \quad \text{sowie} \quad u(0) = g_0,\ u(1) = g_1 \tag{3.2}$$

erfüllt. Dazu diskretisieren wir das Intervall äquidistant mit der Schrittweite $h = \frac{1}{n+1}$, d.h.

$$[0,1] \to \Omega_h = \{x_i = i \cdot h : i = 0, \ldots, n+1\} \tag{3.3}$$

und definieren die abkürzende Schreibweise $u_i := u(x_i) = u(i \cdot h)$. Zur Approximation der zweiten Ableitung betrachten wir die Taylorreihen-Entwicklungen an den Punkten $x + h$ und $x - h$:

$$\begin{aligned} u(x+h) &= u(x) + hu'(x) + \frac{h^2}{2}u''(x) + \frac{h^3}{3!}u'''(x) + \mathcal{O}\left(h^4\right), \\ u(x-h) &= u(x) - hu'(x) + \frac{h^2}{2}u''(x) - \frac{h^3}{3!}u'''(x) + \mathcal{O}\left(h^4\right). \end{aligned}$$

Summieren und Umstellen ergibt

$$u''(x) = \frac{-u(x+h) + 2u(x) - u(x-h)}{h^2} + \mathcal{O}\left(h^2\right)$$

bzw. auf dem Gitter

$$u_i'' = \frac{-u_{i+1} + 2u_i - u_{i-1}}{h^2} + \mathcal{O}\left(h^2\right).$$

Aus (3.2) wird das Gleichungssystem

$$\frac{1}{h^2}\left(-u_{i+1} + 2u_i - u_{i-1}\right) = f_i,\ i = 1, \ldots, n,$$

und $u_0 = g_0$ sowie $u_{n+1} = g_1$, wobei $u_i := u(x_i)$ bzw. $f_i = f(x_i)$ den Funktionswert an den in (3.3) definierten Gitterpunkten symbolisiert. Man erhält in Matrix-Vektor-Schreibweise

$$L_h \boldsymbol{u}_h = \boldsymbol{y}_h \tag{3.4}$$

mit

$$L_h = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix} \tag{3.5}$$

und $y_h(1) = f_1 + h^{-2}g_0$, $y_h(n) = f_n + h^{-2}g_1$ und $y_h(i) = f_i$, $i = 2, \ldots, n-1$.

Direkte Löser (Abschnitt 3.2) lösen das Problem (3.4) sehr effizient. Zur Illustration des Laufzeitverhaltens für höhere Dimensionen ($d \geq 2$) betrachten wir die zwei-dimensionale Poissongleichung

$$-\Delta u = -u_{xx} - u_{yy} = f \text{ in } \Omega \quad \text{und} \quad u = g \text{ auf } \partial\Omega \tag{3.6}$$

auf dem Gebiet $\Omega = [0,1]^2$ mit dem Rand $\partial\Omega = \Omega \backslash (0,1)^2$.

Durch die Diskretisierung entsteht bei lexikografischer Sortierung die $n \times n$-Matrix

$$A_h = \begin{pmatrix} T & -I_{N-1} & & \\ -I_{N-1} & \ddots & \ddots & \\ & \ddots & \ddots & -I_{N-1} \\ & & -I_{N-1} & T \end{pmatrix}, \text{ wobei } T = \begin{pmatrix} 4 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 4 \end{pmatrix} \tag{3.7}$$

und $n = (N-1) \cdot (N-1)$. Wir erkennen, dass sich im Abstand N von der Diagonalen von Null verschiedene Einträge befinden.

3.2 Direkte Verfahren

Es sei A eine Dreiecksmatrix. Dann lässt sich die Lösung von (3.1) im Fall einer unteren Dreiecksmatrix, d.h. $a_{ij} = 0$ für $i < j$, durch Vorwärtssubstitution und im Fall einer oberen Dreiecksmatrix, d.h. $a_{ij} = 0$ für $i > j$, durch Rückwärtssubstitution berechnen. Da A im Allgemeinen keine Dreiecksgestalt hat, wurden die sogenannten *direkten Verfahren* zur (multiplikativen) Zerlegung in Dreiecksmatrizen entwickelt. Sei $A = LU$ eine Zerlegung, wobei L eine untere und U eine obere Dreiecksmatrix sind. Die Lösung von (3.1) bzw. $LU\boldsymbol{x} = \boldsymbol{b}$ ergibt sich durch Lösen der beiden linearen Gleichungssysteme

$$L\boldsymbol{z} = \boldsymbol{b}, \tag{3.8a}$$

$$U\boldsymbol{x} = \boldsymbol{z}. \tag{3.8b}$$

Ein Verfahren zur Berechnung einer Dreieckszerlegung für vollbesetzte und unstrukturierte Matrizen ist die Gauß-Elimination. Ist die Matrix symmetrisch positiv definit, kann die Cholesky-Zerlegung angewendet werden. Die asymptotische Laufzeit direkter Verfahren ist $\mathcal{O}\left(n^3\right)$.

Für schwach besetzte, großdimensionale Matrizen $A \in \mathbb{R}^{n \times n}$ existieren spezielle Versionen der LU-Zerlegung, in der lediglich die Nicht-Nulleinträge in A und die Besetzungsstruktur Berücksichtigung finden. Häufig treten sogenannte *Bandmatrizen* auf, d.h. für Konstanten $p, q \in \mathbb{N}\backslash\{0\}$ verschwinden die Matrixeinträge a_{ij}, falls $i < p+j$ oder $j < i+q$ gilt. Man nennt p die *untere Bandbreite*, q die *obere Bandbreite* und $p+q+1$ die *Bandbreite* der Matrix. Die Komplexität der Gauß-Elimination ist für allgemeine Bandmatrizen $\mathcal{O}\left(npq\right)$.

Die Matrix (3.7) der Diskretisierung des zwei-dimensionalen Modellproblems (3.6) besitzt die minimale Bandbreite $N = \sqrt{n}$. Allgemein ist die minimale Bandbreite für die schwach besetzten FE-Matrizen $n^{1-\frac{1}{d}}$, wobei d die räumliche Dimension des Problems ist.

Während der LU-Faktorisierung werden innerhalb der unteren Bandbreite in L und innerhalb der oberen Bandbreite in U neue Nicht-Null-Einträge generiert, was als *fill-in* bezeichnet wird. Die Beobachtung, dass die Permutation von Zeilen und Spalten die Anzahl der Operationen zur LU-Zerlegung reduzieren kann, führte zur Entwicklung verschiedener Permutations-Strategien, welche das Auffüllen der Faktoren reduzieren. Dies führt dann zu einer effizienteren LU-Faktorisierung.

Der Ansatz von Cuthill und McKee [19] zielt auf eine Reduktion der Bandbreite und folglich des fill-ins ab. Eine Verbesserung des Ansatzes ist das *Reverse Cuthill-McKee*-Verfahren [29].

Die Idee, der von Rose [56] vorgeschlagenen Minimum-Degree-Methode, basiert auf dem Matrixgraph. Die Zeilen und Spalten werden dabei so umgeordnet, dass die Knoten mit kleinem Knotengrad im Eliminationsprozess früher eliminiert werden als Knoten mit großem Knotengrad. Heuristisch wird dadurch das Auffüllen reduziert. Zur Steigerung der Effizienz existieren verschiedene Beschleunigungstechniken, siehe [31]. Da die sukzessive Anpassung der Knotengrade noch nicht eliminierter Knoten aufwändig ist, wurde von Amestoy et al. [1, 2] ein Verfahren zur approximativen Berechnung der Knotengrade entwickelt.

Eine weitere Strategie zur Reduktion des fill-ins ist die Nested-Dissection-Methode [30]. Diese zielt nicht auf die Reduktion der Bandbreite, sondern direkt auf die Reduktion des fill-ins ab. Effiziente Implementierungen nutzen Multilevel Graphzerlegungsmethoden, deren Phasen in Abschnitt 5.3 näher beschrieben werden.

Karypis kommt in [47] beim Vergleich der Umordnungen von Minimum-Degree und von Multilevel-Nested-Dissection bzgl. der Reduktion des fill-ins zu dem Ergebnis, dass die Umordnung nach Multilevel-Nested-Dissection einen substantiell geringeren fill-in hat.

Trotz verschiedener Techniken zur Reduktion des fill-ins bleiben die direkten Verfahren zur Lösung von Gleichungssystemen mit großdimensionalen Bandmatrizen und $d > 2$ unpraktisch. Für FE-Matrizen d-dimensionaler Probleme ist die Komplexität der LU-Faktorisierung mit Nested-Dissection Umordnung $\mathcal{O}\left(n^{3-\frac{3}{d}}\right)$, siehe [52]. Im Fall $d = 3$ wächst die Anzahl der Operationen zur Zerlegung folglich quadratisch.

Im Abschnitt 6.3 werden wir eines der schnellsten direkten Verfahren, PARDISO [59, 60], bzgl. der Laufzeit und des Speicherverbrauchs mit dem vorkonditionierten Verfahren konjugierter Gradienten vergleichen. Dabei kommt der in dieser Arbeit entwickelte Vorkonditionierer zum Einsatz.

3.3 Stationäre iterative Verfahren

Sei M eine reguläre $n \times n$-Matrix. Wird die Zerlegung $A = M + (A - M)$ in das Gleichungssystem (3.1) eingesetzt, so erhalten wir die Fixpunktgleichung

$$\boldsymbol{x} = \underbrace{M^{-1}\boldsymbol{b}}_{=:c} + \underbrace{(I - M^{-1}A)}_{=:T}\boldsymbol{x},$$

aus welcher das Iterationsverfahren

$$\boldsymbol{x}^{(k+1)} = T\boldsymbol{x}^{(k)} + \boldsymbol{c} \tag{3.9}$$

konstruiert wird. Nach dem Banachschen Fixpunktsatz ist (3.9) konvergent, falls T eine Kontraktion ist, denn es gilt

$$\left\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\right\| = \left\|T\boldsymbol{x}^{(k)} - T\boldsymbol{x}^{(k-1)}\right\| \leq \|T\| \cdot \left\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right\|.$$

Ist also $\|T\| < 1$, dann konvergiert das Iterationsverfahren für beliebiges $\boldsymbol{x}^{(0)}$ gegen $A^{-1}\boldsymbol{b}$.

In algorithmischer Schreibweise gilt für die Näherungslösung $\tilde{\boldsymbol{x}} \to T\tilde{\boldsymbol{x}} + \boldsymbol{c}$ und für die exakte Lösung $\boldsymbol{x} = T\boldsymbol{x} + \boldsymbol{c}$. Subtrahiert man den ersten Ausdruck vom zweiten, erhält man:

$$\boldsymbol{e} \to T\boldsymbol{e},$$

mit $\boldsymbol{e} = \boldsymbol{x} - \tilde{\boldsymbol{x}}$. Die rechte Seite $\boldsymbol{b}$ beeinflusst nur den initialen Fehler $\boldsymbol{e}^{(0)} = A^{-1}\boldsymbol{b} - \tilde{\boldsymbol{x}}^{(0)}$. Aus theoretischer Sicht macht es daher keinen Unterschied, das homogene lineare Gleichungssystem $A\boldsymbol{x} = 0$ mit beliebiger Startlösung $\tilde{\boldsymbol{x}}^{(0)}$ oder das inhomogene lineare Gleichungssystem $A\boldsymbol{x} = \boldsymbol{b}$ zu analysieren. Die Kenntnis der Lösung des homogenen Systems kann die Analyse jedoch vereinfachen.

Zur Definition konkreter stationärer iterativer Verfahren wird die Matrix oft zerlegt. Sei $A = A_L + A_D + A_R$, wobei A_D der Diagonalanteil, A_L der strikte untere Anteil und A_R der strikte obere Anteil von A seien. Für die Wahl $M = A_D$ ergibt sich das Gesamtschritt-Verfahren (Jacobi-Verfahren).

3.3.1 Konvergenz des Jacobi-Verfahrens für das Modellproblem

Die Matrix des eindimensionalen Modellproblems (3.4) besitzt die Struktur einer tridiagonalen Matrix. Die Eigenwerte der tridiagonalen Matrix $A = \text{tridiag}(b, a, b) \in \mathbb{R}^{n\times n}$ sind

$$\lambda_\ell = a + 2b\cos\left(\frac{\pi\ell}{n+1}\right), \quad \ell = 1, \ldots, n,$$

und die zugehörigen Eigenvektoren lauten

$$(\boldsymbol{s}^{(\ell)})_j = \sin\left(\frac{\pi j\ell}{n+1}\right), \quad j = 1, \ldots, n. \tag{3.10}$$

Wenden wir das Jacobi-Verfahren auf (3.4) an, so ist $M = \text{diag}(L_h)$ und die Inverse $M^{-1} = h^2\,\text{diag}(\frac{1}{2})$ kann leicht gebildet werden.

Allgemein lautet der Fehler im k-ten Iterationsschritt

$$\begin{aligned}\boldsymbol{e}^{(k)} &= \boldsymbol{x} - \boldsymbol{x}^{(k)} = \boldsymbol{x} - M^{-1}\boldsymbol{b} - (I - M^{-1}A)\boldsymbol{x}^{(k-1)} \\ &= \boldsymbol{x} - \boldsymbol{x}^{(k-1)} - M^{-1}A\boldsymbol{x} + M^{-1}A\boldsymbol{x}^{(k-1)} = (I - M^{-1}A)\boldsymbol{e}^{(k-1)}\end{aligned}$$

und für das Jacobi-Verfahren angewendet auf (3.4) gilt

$$\boldsymbol{e}^{(k)} = \left(I - \frac{h^2}{2}L_h\right)\boldsymbol{e}^{(k-1)}.$$

Wir stellen den Fehler in der durch die (orthogonalen) Eigenvektoren (3.10) gebildeten Basis dar:

$$\begin{aligned}\boldsymbol{e}^{(k)} = \sum_{\ell=1}^{n}\alpha_\ell^{(k)}\boldsymbol{s}^{(\ell)} = \left(I - \frac{h^2}{2}L_h\right)\boldsymbol{e}^{(k-1)} &= \sum_{\ell=1}^{n}\alpha_\ell^{(k-1)}\left(\boldsymbol{s}^{(\ell)} - \frac{h^2}{2}L_h\boldsymbol{s}^{(\ell)}\right) \\ &= \sum_{\ell=1}^{n}\alpha_\ell^{(k-1)}\left(\boldsymbol{s}^{(\ell)} - \frac{h^2}{2}\lambda_\ell\boldsymbol{s}^{(\ell)}\right) = \sum_{\ell=1}^{n}\alpha_\ell^{(k-1)}\left(1 - \frac{h^2}{2}\lambda_\ell\right)\boldsymbol{s}^{(\ell)}.\end{aligned}$$

Aus der Eindeutigkeit der Darstellung folgt mit $\lambda_\ell = \frac{1}{h^2}\left(2 - 2\cos\frac{\pi\ell}{n+1}\right)$

$$\alpha_\ell^{(k)} = \alpha_\ell^{(k-1)}\left(1 - \frac{h^2}{2}\lambda_\ell\right) = \alpha_\ell^{(k-1)}\cos\left(\frac{\pi\ell}{n+1}\right).$$

D.h. der Beitrag des ℓ-ten Eigenvektors zum Fehler wird mit dem Faktor $\cos\frac{\pi\ell}{n+1}$ gedämpft. Da die Kosinusfunktion bei $\frac{\pi}{2}$ eine Nullstelle besitzt, werden die Fehleranteile mit Frequenzen in der Nähe der Frequenz $2\ell \approx n+1$ stark gedämpft. Außerhalb des mittleren Frequenzbereichs findet nur schwache Dämpfung statt.

Besteht der Startvektor aus nieder- und hochfrequenten Anteilen, wie in Abb. 3.1a dargestellt, so wird die Amplitude der hochfrequenten Schwingung nach 64 Iterationen zwar deutlich reduziert, lässt sich aber noch klar erkennen, siehe Abb. 3.1c. Man sieht auch, dass der niederfrequente Fehleranteil nur ungenügend behandelt wird.

In der Abb. 3.1b ist der Startvektor für das Jacobi-Verfahren zur Lösung des Modellproblems aus einer nieder- und mittelfrequenten Schwingung zusammengesetzt. Der mittelfrequente Anteil des Fehler ist nach 64 Iterationen fast nicht mehr erkennbar, während der niederfrequente Anteil sich nicht merklich reduziert hat, siehe Abb. 3.1d.

Weil das Jacobi-Verfahren nicht global konvergiert, kann es als eigenständiges Verfahren nur eingeschränkt genutzt werden.

3.3.2 Relaxiertes Jacobi-Verfahren für das Modellproblem

Durch die Wahl einfacher, von einem Parameter ω abhängiger, Matrizen $M(\omega)$ sollen die Konvergenzeigenschaften verändert werden. Setzen wir

$$M(\omega) = \frac{1}{\omega}A_D, \tag{3.11}$$

erhalten wir ein relaxiertes Jacobi-Verfahren

$$x_i^{(k+1)} = (1-\omega)x_i^{(k)} + \frac{\omega}{a_{ii}}\left(b_i - \sum_{i\neq j} a_{ij}x_j(k)\right), \quad i = 1,\ldots,n.$$

Für das Modellproblem (3.4) werden wir sehen, dass der Frequenzbereich in dem Dämpfung auftritt, durch die Wahl von ω beeinflusst werden kann. Jedoch wird die Dämpfung nicht für alle Fehleranteile gleichzeitig gelingen.

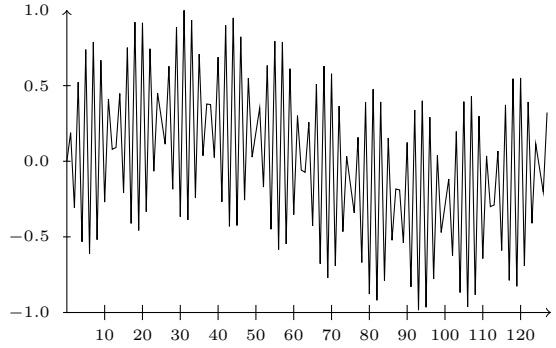
$$M(\omega) = \frac{1}{h^2\omega}\operatorname{diag}(2) \quad \Leftrightarrow \quad M(\omega)^{-1} = \frac{\omega h^2}{2}I$$

Im relaxierten Jacobi-Verfahren lautet der Fehler im k-ten Iterationsschritt

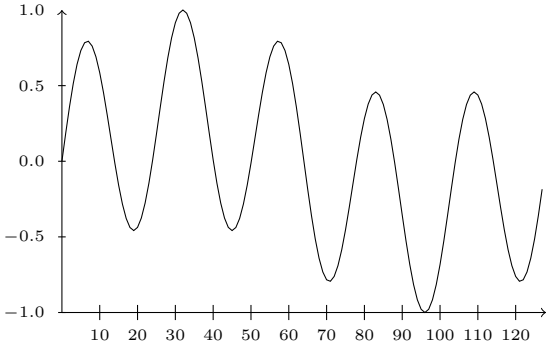
$$\boldsymbol{e}^{(k)} = \left(I - \frac{\omega h^2}{2}L_h\right)\boldsymbol{e}^{(k-1)}.$$

Wie im vorigen Abschnitt ergibt sich

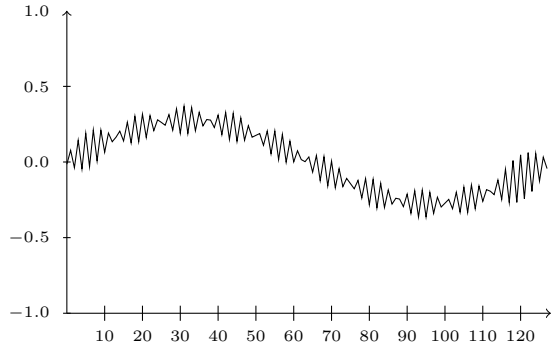
$$\boldsymbol{e}^{(k)} = \sum_{\ell=1}^{n}\alpha_\ell^{(k)}\boldsymbol{s}^{(\ell)} = \sum_{\ell=1}^{n}\alpha_\ell^{(k-1)}\left(1 - \frac{\omega h^2}{2}\lambda_\ell\right)\boldsymbol{s}^{(\ell)}$$



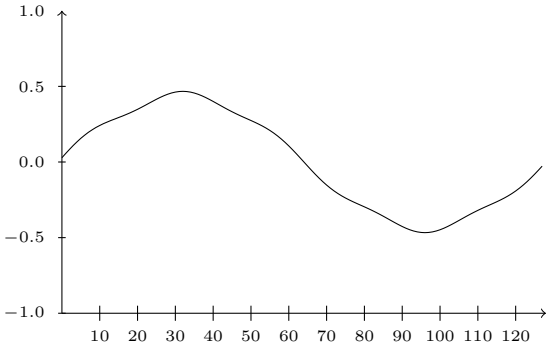
(a) Überlagerung eines nieder- und hochfrequenten Fehlers



(b) Überlagerung eines nieder- und mittelfrequenten Fehlers



(c) Entwicklung des Fehlers aus Abb. 3.1a nach 64 Iterationen



(d) Entwicklung des Fehlers aus Abb. 3.1b nach 64 Iterationen

Abb. 3.1: Dämpfung des Jacobi-Verfahrens

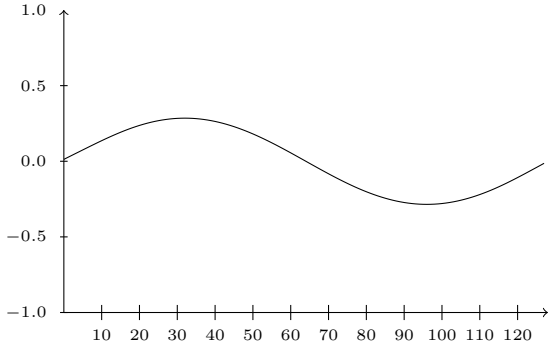
bzw.

$$\alpha_\ell^{(k)} = \alpha_\ell^{(k-1)} \left(1 - \omega + \omega \cos\left(\frac{\pi \ell}{n+1}\right)\right).$$

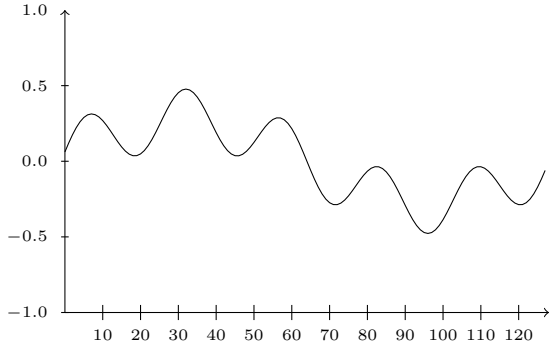
Wir wollen versuchen, ω so zu bestimmen, dass

$$\max_{\ell \in \{1,\dots,n\}} \left|1 - \omega + \omega \cos\left(\frac{\pi \ell}{n+1}\right)\right| \to \min$$

gilt. Wählen wir $\omega = \frac{2}{3}$, werden höhere Frequenzen gedämpft (siehe Abb. 3.3). Im Vergleich zum Jacobi-Verfahren werden höhere Frequenzen besser gedämpft, siehe Abb. 3.2a bzw. 3.1c. Die Elimination mittlerer Frequenzen gelingt mit dem Relaxationsverfahren im Vergleich zum unrelaxierten Jacobi-Verfahren dagegen nicht so gut, siehe Abb. 3.2b bzw. 3.1d. Den beobachteten Effekt bezeichnet man als *Glättung*, da hochfrequente Fehleranteile verschwinden (siehe Abb. 3.2).



(a) Entwicklung des Fehlers aus Abb. 3.1a nach 64 Iterationen



(b) Entwicklung des Fehlers aus Abb. 3.1b nach 64 Iterationen

Abb. 3.2: Dämpfung der Jacobi-Relaxation für $\omega = \frac{2}{3}$

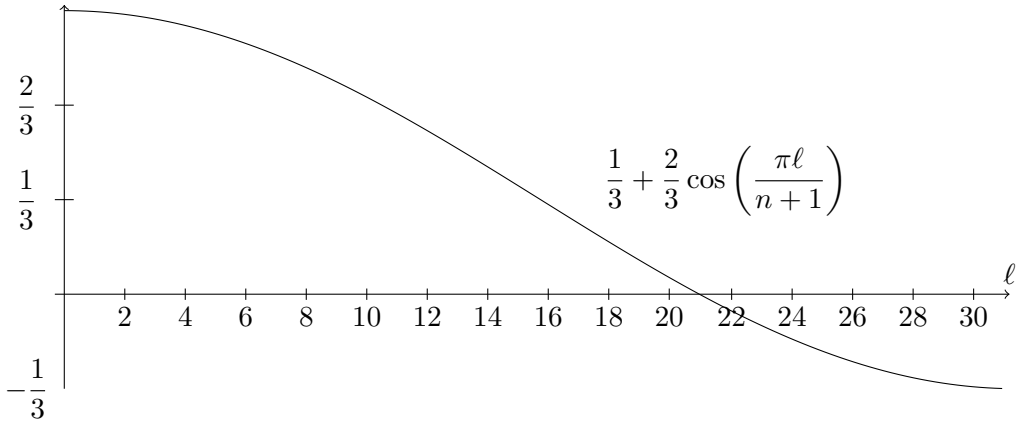


Abb. 3.3: höhere Frequenzen werden gedämpft

Weitere stationäre Iterationsverfahren mit Glättungseigenschaften sind zum Beispiel das Gauss-Seidel-Verfahren, das SOR-Verfahren, SSOR-Verfahren oder das Richardson-Verfahren, siehe [36].

3.4 Geometrische und algebraische Mehrgitterverfahren

Mit wachsender Problemgröße n werden (iterative) Verfahren benötigt, deren Konvergenzrate unabhängig von n ist. Fedorenko [25, 26] wandte in den 1960er Jahren ein Mehrgitterverfahren auf die durch das Finite-Differenzen-Verfahren diskretisierte Poissongleichung an. Bakhvalov [5] verallgemeinerte die Idee auf elliptische Operatoren mit variablen Koeffizienten. Die Arbeiten von Brandt [17] und Hackbusch [35] steigerten die Bekanntheit der Mehrgitterverfahren, die eine große Klasse von Problemen mit linearem Aufwand lösen.

3.4.1 Komponenten der Mehrgitterverfahren

Mehrgitterverfahren nutzen nicht nur eine Diskretisierung des Problems, sondern eine Hierarchie von Diskretisierungen, um die Glättungseigenschaften von Relaxationsverfahren besonders gut ausnutzen zu können. Basiert die Hierarchie auf geometrischen Informationen, so nennt man die Methode auch *geometrisches Mehrgitterverfahren*.

Wir erläutern die grundlegenden Ideen von Mehrgitterverfahren anhand eines (geometrischen) Zweigitterverfahrens. Dazu betrachten wir zwei Diskretisierungen Ω_h, Ω_H des Gebietes Ω des Modellproblems der Art (3.3) mit Schrittweiten $h < H$. Wir bezeichnen Ω_h als *feines Gitter* und Ω_H als *grobes Gitter*.

Wie wir in Abschnitt 3.3 am Modellproblem gesehen haben, können stationäre iterative Verfahren als Glätter wirken. Die auf Ω_h nur ungenügend reduzierten niederfrequenten Fehleranteile lassen sich auf dem groben Gitter gut approximieren und entsprechen auf Ω_H vergleichsweise hohen Frequenzen. Diese können durch die Anwendung eines Relaxationsverfahrens effektiv behandelt werden.

Der Glätter S_h wird ν_1-mal auf das Gleichungssystem

$$A_h \boldsymbol{v}^h = \boldsymbol{b}^h \tag{3.12}$$

angewendet, wobei wir einen Startvektor $\boldsymbol{u}^h$ wählen. Daraus resultiert das Residuum

$$\boldsymbol{r}^h = \boldsymbol{b}^h - A_h \boldsymbol{v}^h. \tag{3.13}$$

Um das Residuum zu korrigieren, wird (3.13) zunächst auf das grobe Gitter übertragen, was man als *Restriktion* $I_h^H : \Omega_h \to \Omega_H$ bezeichnet. Offensichtlich reduziert sich dabei die Anzahl der Unbekannten. Das dem groben Gitter zugeordnete Gleichungssystem

$$A_H \boldsymbol{e}^H = \boldsymbol{r}^H := I_h^H \boldsymbol{r}^h \tag{3.14}$$

kann mit geringerem Aufwand als das ursprüngliche System gelöst werden.

Durch die sogenannte *Prolongation* bzw. *Interpolation* $I_H^h : \Omega_H \to \Omega_h$ wird die Lösung $\boldsymbol{e}^H$ von (3.14) zurück auf das feine Gitter übertragen, $\boldsymbol{e}^h := I_H^h \boldsymbol{e}^H$. Eine bessere Näherung an die Lösung von (3.12) ergibt sich durch die Korrektur $\boldsymbol{v}^h + \boldsymbol{e}^h$. Um die bei der Interpolation im Allgemeinen auftretenden oszillierenden Fehler zu eliminieren, wird ein Glätter S_h ν_2-mal auf die korrigierte Gleichung angewendet. Alg. 3 fasst das Verfahren zusammen.

Algorithmus 3 Zweigitter-Schema

1: wende Relaxationsverfahren ν_1-mal auf $A_h \boldsymbol{v}^h = \boldsymbol{b}^h$ an ($\boldsymbol{u}^h$ Startvektor)
2: restringiere das Residuum $\boldsymbol{r}^h = \boldsymbol{b}^h - A_h \boldsymbol{v}^h$ auf das grobe Gitter: $\boldsymbol{r}^H = I_h^H \boldsymbol{r}^h$
3: löse $A_H \boldsymbol{e}^H = \boldsymbol{r}^H$
4: prolongiere den Fehler $\boldsymbol{e}^H$: $\boldsymbol{e}^h = I_H^h \boldsymbol{e}^H$
5: korrigiere die Feingitterapproximation: $\boldsymbol{v}^h + \boldsymbol{e}^h \to \boldsymbol{v}^h$
6: wende Relaxationsverfahren ν_2-mal auf $A_h \boldsymbol{u}^h = \boldsymbol{b}^h$ an ($\boldsymbol{v}^h$ Startvektor)

Aus dem Zweigitter-Schema wird ein Mehrgitteralgorithmus, falls die Lösung von $A_H \boldsymbol{e}^H = \boldsymbol{r}^H$ durch rekursive Anwendung des Alg. 3 gewonnen wird.

Zur vollständigen Spezifikation des Algorithmus müssen die Restriktion und Prolongation definiert werden, die bei geometrischen Mehrgitterverfahren aus der Gitterhierarchie abgeleitet werden.

Das geometrische Mehrgitterverfahren ist bei entsprechender Anpassung an das jeweilige Problem sehr effizient. Ein Nachteil besteht im relativ hohen Implementierungsaufwand für nicht triviale Geometrien, denn es müssen die geometrischen Informationen verwaltet und entsprechend die Restriktionen und Prolongationen definiert werden. Unstetige Koeffizienten im Differentialoperator oder Anisotropien müssen ebenfalls bei der Definition der Interpolation bzw. Restriktion beachtet werden.

3.4.2 Algebraische Mehrgitterverfahren

Die Abhängigkeit der geometrischen Mehrgitterverfahren von der Geometrie soll durch die Einführung algebraischer Mehrgitterverfahren (algebraic multigrid, AMG) beseitigt werden. Eine grundlegende Arbeit zu AMG stammt von Ruge und Stüben [57]. Das dort vorgestellte Verfahren wird auch klassisches AMG genannt. Unsere numerischen Ergebnisse werden wir mit BoomerAMG [44], einer effizienten Implementierung vom AMG, vergleichen.

Einen Ersatz für geometrische Gitter bietet der (gewichtete) Matrixgraph G_{A_h}. In Analogie zu klassischen Mehrgitterverfahren werden die Knoten des Matrixgraphen als Punkte bezeichnet. Die Punkte und Kanten bilden dann eine Art Gitter. Im Gegensatz zu geometrischen Mehrgitterverfahren, bei denen die Restriktion und Prolongation von vornherein festgelegt und die Glätter entsprechend gewählt werden, ist die Glättung bei algebraischen Mehrgitterverfahren fest und Restriktion sowie Prolongation werden durch das Ausnutzen der Glattheitseigenschaften des Fehlers gewonnen.

Im klassischen algebraischen Mehrgitterverfahren besteht die Berechnung der Restriktion bzw. Interpolation aus drei Schritten:

1. der Wahl einer Zerlegung der Punkte des Matrixgraphen in Grobgitterpunkte C^h und Feingitterpunkte F^h, wobei $\Omega_H := C^h$,

2. der Definition von Interpolationsgewichten w_{ik}^h zur Interpolation

$$(I_H^h \boldsymbol{e}^H)_i = \sum_{k \in C^h} w_{ik} \boldsymbol{e}_k^H \quad (i \in \Omega^h) \quad \text{mit } w_{ik} = \delta_{ik} \quad \text{falls } i \in C^h \tag{3.15}$$

   und

3. der Berechnung der Restriktion $I_h^H = \left(I_H^h\right)^T$ sowie der Grobgittermatrix $A_H = I_h^H A_h I_H^h$.

In (3.15) werden lediglich die Interpolationsgewichte für die Grobgitterpunkte beschrieben. Eine mögliche Spezifikation der Gewichte für die Feingitterpunkte erfolgt weiter unten.

Zur Auswahl der C^h- und F^h-Punkte existieren zahlreiche Heuristiken, wie Multi-Coloring und Umordnung bzgl. unabhängiger Mengen. Wird bei der Auswahl der C^h- und

F^h-Punkte die physikalische Bedeutung berücksichtigt, spricht man von einem *unknown based approach* [57].

Die Matrix A_H und die rechte Seite $\boldsymbol{r}^H$ des groben Gitters in Zeile 3 des Alg. 3 können folgendermassen erzeugt werden:

$$A_H = I_h^H A_h I_H^h, \quad \boldsymbol{r}^H = I_h^H \boldsymbol{r}^h. \tag{3.16}$$

Konvergenz. Ein Fehler $\boldsymbol{e}^h$ wird *algebraisch glatt* genannt, falls $\boldsymbol{e}^h$ durch den Glätter nicht effizient behandelt werden kann, d.h. $\left\|S_h \boldsymbol{e}^h\right\|_{A_h} \approx \left\|\boldsymbol{e}^h\right\|_{A_h}$. Die Restriktion muss dann so definiert werden, dass die durch die Grobgitterkorrektur effizient reduzierbaren Fehleranteile auch auf das grobe Gitter übertragen werden.

Für das Gauss-Seidel-Verfahren und das Jacobi-Verfahren gilt nach [57] die *Glättungseigenschaft*

$$\left\|S_h \boldsymbol{e}^h\right\|_{A_h}^2 \leq \left\|\boldsymbol{e}^h\right\|_{A_h}^2 - \alpha \left\|A_h \boldsymbol{e}^h\right\|_{D^{-1}}^2 \quad \forall \boldsymbol{e}^h \in \Omega_h, \tag{3.17}$$

für ein $\alpha > 0$ und $D = \operatorname{diag} A_h$.

Zur Analyse des Alg. 3 setzen wir $\boldsymbol{b}^h = 0$. Dann kann Alg. 3 als ein Iterationsprozess der Form (3.9) interpretiert werden

$$\boldsymbol{u}_{\text{neu}}^h = S_h^{\nu_2} \left[S_h^{\nu_1} \boldsymbol{u}_0^h + I_H^h A_H^{-1} I_h^H \left(-A_h S_h^{\nu_1} \boldsymbol{u}_0^h \right) \right].$$

Für die Iterationsmatrix T aus (3.9) ergibt sich der Ausdruck

$$T = S_h^{\nu_2} \left[I - I_H^h A_H^{-1} I_h^H A_h \right] S_h^{\nu_1},$$

wobei der Teil

$$T_h^H = I - I_H^h A_H^{-1} I_h^H A_h \tag{3.18}$$

als *Grobgitterkorrektur* bezeichnet wird. Wichtige Eigenschaften von (3.18) fasst das folgende Lemma zusammen.

Lemma 3.1. *Wenn die Grobgittermatrix durch* (3.16) *definiert wird, dann ist die Grobgitterkorrektur* (3.18) *eine Projektion bzgl. des A_h-Skalarprodukts. Der Bildraum von T_h^H ist A_h-orthogonal zum Bildraum von I_h^H.*

Beweis. siehe [58, Lemma 13.1] □

Im Folgenden betrachten wir nur den Fall der Nachglättung. Setzen wir die Beziehung

$$\left\|S_h \boldsymbol{e}^h\right\|_{A_h} = \sqrt{1-\varepsilon} \left\|\boldsymbol{e}^h\right\|_{A_h}, \quad 0 < \varepsilon \ll 1,$$

für algebraisch glatte Fehler in die Glättungseigenschaft (3.17) ein, so ergibt sich durch Umstellen

$$\left\|\boldsymbol{r}^h\right\|_{D^{-1}}^2 \leq \frac{\varepsilon}{\alpha} \left\|\boldsymbol{e}^h\right\|_{A_h}^2. \tag{3.19}$$

D.h. die Fehlerglättung ist ineffizient, falls die Norm des Residuums $\left\|\boldsymbol{r}^h\right\|_{D^{-1}}$ klein gegenüber der Norm des Fehlers $\left\|\boldsymbol{e}^h\right\|_{A_h}$ wird. Folglich sollte die Grobgitterkorrektur T_h^H die

Norm $\left\|\boldsymbol{r}^h\right\|_{D^{-1}}$ relativ zur Norm $\left\|\boldsymbol{e}^h\right\|_{A_h}$ erhöhen. Wir benötigen also eine Ungleichung der folgenden Form

$$\left\|T_h^H \boldsymbol{e}^h\right\|_{A_h}^2 \leq \beta \left\|A_h T_h^H \boldsymbol{e}^h\right\|_{D^{-1}}^2, \quad \beta > 0 \tag{3.20}$$

mit $\boldsymbol{e}^h \in \text{range}(T_h^H)$. Die Bedingung (3.20) lässt sich aus der Bedingung

$$\min_{\boldsymbol{e}^H \in \Omega_H} \left\|\boldsymbol{e}^h - I_H^h \boldsymbol{e}^H\right\|_D^2 \leq \beta \left\|\boldsymbol{e}^h\right\|_{A_h}^2, \quad \beta > 0, \tag{3.21}$$

ableiten.

Lemma 3.2. *Sei A_h symmetrisch positiv definit, der Prolongationsoperator I_H^h habe vollen Rang und der Restriktionsoperator sei $I_h^H = (I_H^h)^T$. Außerdem soll die Glättungseigenschaft* (3.17) *mit $\alpha > 0$ und die Approximationseigenschaft* (3.21) *erfüllt sein. Dann konvergiert das Zweigitter-Verfahren für $\alpha \leq \beta$ und der Zweigitter-Konvergenzfaktor*

$$\left\|S_h T_h^H\right\|_{A_h} \leq \sqrt{1 - \frac{\alpha}{\beta}}$$

ist beschränkt.

Beweis. siehe [57, Theorem 5.2] □

Bemerkung 3.3. Man kann die Konvergenz durch eine große Anzahl an Grobgitterpunkten erzwingen. Der Aufwand pro Gitterebene ist dabei aber sehr hoch.

Praktische Umsetzung. Konkrete algebraische Mehrgitter-Algorithmen bestehen meist aus zwei Phasen:

1. Der *Setup-Phase*, die die Definition des groben Gitters sowie der zugehörigen Restriktion bzw. Prolongation und die Definition des Grobgitterkorrekturoperators umfasst.

2. Der *Lösungsphase*, in der die in der ersten Phase berechneten Komponenten in Alg. 3 eingesetzt und das lineare System gelöst werden.

Algorithmus 4 Setup-Phase

1: $m = 1$
2: **while** A^m noch nicht leicht invertierbar **do**
3: paritioniere $\Omega^m = C^m \cup F^m$ und definiere Interpolation I_{m+1}^m, $\Omega^{m+1} = C^m$
4: $I_m^{m+1} = \left(I_{m+1}^m\right)^T$, $A^{m+1} = I_m^{m+1} A^m I_{m+1}^m$
5: $m = m + 1$
6: $q = m - 1$

Um den noch nicht genau spezifizierten Übergang vom feinen auf das grobe Gitter zu definieren, wird das Gleichungssystem

$$A^m \boldsymbol{u}^m = \boldsymbol{b}^m$$

betrachtet. Zur einfacheren Erklärung nehmen wir an, dass A^1 eine symmetrische M-Matrix[1] ist. Im Algorithmus selbst wird dies nicht vorausgesetzt.

Aus (3.19) folgt für algebraisch glatte Fehler im Mittel

$$a_{ii}^m e_i^m \approx - \sum_{j \in N^m(i)} a_{ij}^m e_j^m, \quad i \in \Omega^m, \tag{3.22}$$

wobei $N^m(i)$ die Nachbarn im Matrixgraphen G_{A^m} sind.

Wir nehmen an, dass der Interpolationsoperator wie in (3.15) konstruiert wird. Dann gilt

$$(I_{m+1}^m \boldsymbol{e}^{m+1})_i = \begin{cases} e_i^{m+1} & \text{falls } i \in C^m, \\ \sum_{k \in C^m(i)} w_{ik} e_k^{m+1} & \text{falls } i \in F^m, \end{cases}$$

wobei $C_m(i)$ eine kleine Teilmenge der Grobgitterpunkte sei (um die Interpolation lokal zu halten).

Wählt man beispielweise C^m und F^m so, dass die Nachbarn für alle $i \in F^m$ in C^m liegen, d.h. $N^m(i) \subset C^m$ und $C^m(i) = N^m(i)$ dann kann (3.22) direkt zur Bestimmung der Interpolationsgewichte genutzt werden,

$$w_{ik}^m = -\frac{a_{ik}^m}{a_{ii}^m}, \quad i \in F^m, \ k \in C^m.$$

Wie in Bemerkung 3.3 dargelegt, ist dies meist unpraktisch.

Zur Reduktion des Aufwands soll der Wert für $i \in F^m$ aus möglichst wenig Punkten $C^m(i)$ interpoliert werden. Dann muss für alle $j \in F^m(i) = N^m(i) \backslash C^m(i)$ der Term e_j^m in der Interpolationsformel (3.22) durch Terme von e_k^m, $k \in C^m(i)$ approximiert werden.

In (3.22) wird e_i^m wesentlich durch die e_j^m bestimmt, für die $|a_{ij}|$ groß ist. Dies motiviert die Definition der starken Verbindungen. Ein Punkt i ist mit dem Punkt j *stark verbunden*, falls

$$-a_{ij} \geq \Theta \max_{\ell \neq i} \{-a_{i\ell}\}, \quad 0 < \Theta \leq 1$$

ist. Üblicherweise wird $\Theta = 0.25$ gewählt. Wir setzen

$$S^m(i) := \{j : i \text{ ist stark verbunden mit } j\} \quad \text{und} \quad C^m(i) = C^m \cap S^m(i).$$

Zudem seien $F_s^m(i) := F^m(i) \cap S(i)$ und $F_w^m(i) = F^m(i) \backslash F_s^m(i)$. Die Formel (3.22) lässt sich mit diesen Bezeichnungen schreiben:

$$a_{ii}^m e_i^m \approx - \sum_{j \in C^m(i)} a_{ij}^m e_j^m - \sum_{j \in F_s^m(i)} a_{ij}^m e_j^m - \sum_{j \in F_w^m(i)} a_{ij}^m e_j^m.$$

[1] Eine Matrix $A = sI - B$, mit $b_{ij} \geq 0$ für alle i, j und $s > 0$, für die $s \geq \rho(B)$ gilt, nennt man *M-Matrix*, siehe [62].

Anstatt die Summe der Terme für $j \in F_w^m(i)$ wegzulassen, kann diese zum Diagonalanteil addiert werden[2]:

$$\left(a_{ii}^m + \sum_{j \in F_w^m(i)} a_{ij}^m \right) e_i^m \approx - \sum_{j \in C^m(i)} a_{ij}^m e_j^m - \sum_{j \in F_s^m(i)} a_{ij}^m e_j^m .$$

Punkte $j \in F_s^m(i)$ können nicht einfach zur Diagonalen addiert werden. Die Komponenten e_j^m werden jedoch mit (3.22) ebenso durch den Fehlervektor $\boldsymbol{e}^m$ interpoliert wie e_i^m. Die Komponenten e_j^m, $j \in F_s^m(i)$ werden durch

$$e_j^m \approx \frac{\sum_{\ell \in C^m(i)} a_{j\ell}^m e_\ell^m}{\sum_{\ell \in C^m(i)} a_{j\ell}^m} \tag{3.23}$$

approximiert. Einsetzen in (3.22) liefert dann

$$\left(a_{ii}^m + \sum_{j \in F_w^m(i)} a_{ij}^m \right) e_i^m \approx - \sum_{j \in C^m(i)} a_{ij}^m e_j^m - \sum_{j \in F_s^m(i)} a_{ij}^m \left(\frac{\sum_{\ell \in C^m(i)} a_{j\ell}^m e_\ell^m}{\sum_{\ell \in C^m(i)} a_{j\ell}^m} \right) .$$

Setzt man $\delta_j = \sum_{\ell \in C^m(i)} a_{j\ell}^m$, so ergibt sich der interpolierte Wert an der Stelle i aus

$$e_i^m = \sum_{k \in C^m(i)} w_{ik}^m e_k^{m+1} \quad \text{wobei} \quad w_{ik}^m = - \frac{a_{ik}^m + \sum_{j \in F_s^m(i)} \frac{a_{ij}^m a_{jk}^m}{\delta_j}}{a_{ii}^m + \sum_{j \in F_w^m(i)} a_{ik}^m} .$$

Je mehr starke Verbindungen die j-te Komponente von $\boldsymbol{e}^m$ in die Menge $C^m(i)$ besitzt, desto besser ist die Approximation (3.23). Aus der Beobachtung, dass in der Praxis bereits eine starke Verbindung ausreichend ist, wird in [57] ein Kriterium zur Wahl der Grobgitterpunkte abgeleitet.

Kriterium 1 Für jedes $i \in F^m$ sollte jeder Punkt $j \in S(i)$ entweder in C^m liegen oder mit zumindest einem Punkt in $C^m(i) = C^m \cap S(i)$ stark verbunden sein.

Die Mehrgitterkomponenten werden mit Blick auf einen effizienten Lösungsprozess definiert. Die Effizienz wird einerseits durch den Konvergenzfaktor bestimmt, andererseits durch den Aufwand zur Berechnung der Lösung auf den einzelnen Gittern. Das bedeutet: Je kleiner die Mengen C^m und daher die Gitter auf den Stufen $m = 1, \ldots, q$ werden, desto weniger Aufwand zur Lösung fällt an. Deshalb wird ein weiteres Kriterium zur Berechnung der Interpolation eingeführt.

Kriterium 2 Die Menge C^m soll eine maximale Teilmenge von Ω^m sein, so dass keine zwei Grobgitterpunkte stark verbunden sind.

AMG ist sehr effizient zur Lösung von (3.1) für (genau) eine rechte Seite. Ändert sich die rechte Seite $\boldsymbol{b}$ in (3.1), müssen die Stufen neu berechnet werden, da $\boldsymbol{b}$ in die Konstruktion der Stufen involviert ist.

[2]Kompensationsstrategie wird auch bei ILU-Verfahren eingesetzt

3.5 Das Konjugierte-Gradienten-Verfahren

Krylov-Raum-Verfahren gehören aktuell zu den wichtigsten iterativen Verfahren zur Lösung großdimensionaler, schwach besetzter, linearer Gleichungssysteme. Das Konjugierte-Gradienten-Verfahren (englisch conjugate gradient method, kurz CG-Verfahren) ist ein Krylov-Raum-Verfahren für symmetrisch positiv definite Gleichungssysteme.

Im Gegensatz zu stationären Iterationsverfahren werden die Iterierten nicht komponentenweise aktualisiert. Daher werden im Alg. 5 die i-te Iterierte jetzt mit $\boldsymbol{x}_i$, das Residuum bzw. der Fehler im i-ten Schritt mit $\boldsymbol{r}_i$ bzw. $\boldsymbol{e}_i$ bezeichnet. Die i-te Suchrichtung bezeichnen wir mit $\boldsymbol{d}_i$.

Algorithmus 5 CG-Verfahren

$i = 0$
$\boldsymbol{d}_i = \boldsymbol{r}_i = \boldsymbol{b} - A\boldsymbol{x}_i$, $\varrho_i = \langle \boldsymbol{r}_i | \boldsymbol{r}_i \rangle$
while $\|\boldsymbol{r}_i\| > \varepsilon$ und $i < i_{\max}$ **do**
    $\boldsymbol{w}_i = A\boldsymbol{d}_i$
    $\alpha_i = \dfrac{\varrho_i}{\langle \boldsymbol{d}_i | \boldsymbol{w}_i \rangle}$
    $\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{d}_i$
    $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i \boldsymbol{w}_i$
    $\varrho_{i+1} = \langle \boldsymbol{r}_{i+1} | \boldsymbol{r}_{i+1} \rangle$
    $\beta_{i+1} = \dfrac{\varrho_{i+1}}{\varrho_i}$
    $\boldsymbol{d}_{i+1} = \boldsymbol{r}_{i+1} + \beta_{i+1} \boldsymbol{d}_i$
    $i = i + 1$

3.5.1 Konvergenz des Konjugierte-Gradienten-Verfahrens

Die Konvergenzgeschwindigkeit des CG-Verfahrens hängt vom Spektrum bzw. der Kondition der Matrix ab.

Satz 3.4. *Für eine symmetrisch positiv definite Matrix A konvergiert das CG-Verfahren mit der Konvergenzabschätzung*

$$\|\boldsymbol{x}_{i+1} - \boldsymbol{x}\|_A \leq \frac{2q^i}{1+q^{2i}} \|\boldsymbol{e}_0\|_A \tag{3.24}$$

und

$$q = \frac{\sqrt{\kappa_2(A)} - 1}{\sqrt{\kappa_2(A)} + 1}, \quad \kappa_2(A) = \|A\|_2 \left\|A^{-1}\right\|_2 = \frac{\lambda_{\max}(A)}{\lambda_{\min}(A)}.$$

Beweis. siehe [61] □

Ist die spektrale Kondition der Matrix A groß, d.h. das Problem ist schlecht konditioniert, so liegt q in der Nähe von Eins und das CG-Verfahren konvergiert nur langsam.

Im Abschnitt 3.6 werden Verfahren zur Verbesserung der Konvergenzrate skizziert. Weitere Konvergenzaussagen mit Beweisen sind beispielsweise in [32, 36, 58].

3.6 Vorkonditionierer

„Finding a good preconditioner to solve a given sparse linear system is often viewed as a combination of art and science.“ (Yousef Saad, [58, Kapitel 10])

Im Abschnitt 3.3 wurde anhand des Modellproblems deutlich, dass die Konvergenz stationärer iterativer Verfahren vom Spektrum der Matrix abhängt. Obwohl das CG-Verfahren aus Abschnitt 3.5 in exakter Arithmetik nach endlich vielen Iterationen das exakte Ergebnis liefert, ist die Anzahl der Schritte bis zur Konvergenz für großdimensionale Probleme im Allgemeinen groß. Deshalb sind Vorkonditionierer für den erfolgreichen Einsatz iterativer Methoden wichtig. Wie der Name bereits andeutet, wird eine Transformation angewendet, welche die Kondition des ursprünglichen Systems verändert. Ziel der Transformation ist es, die Effizienz und Robustheit iterativer Verfahren zu erhöhen.

Aus der linearen Algebra ist bekannt, dass die Multiplikation eines Gleichungssystems (3.1) mit einer regulären Matrix C zwar die Kondition, aber nicht die Lösung des Problems ändert. Aus praktischer Sicht werden gewöhnlich zwei Forderungen an die Matrix C gestellt:

1. das Gleichungssystem $C\tilde{\boldsymbol{x}} = \boldsymbol{x}$ soll leicht zu lösen sein und
2. durch die Multiplikation mit C soll die Kondition des Problems so verändert werden, dass das Lösungsverfahren schnell konvergiert.

Sei C eine Matrix, die den beiden Forderungen genügt. Die Multiplikation von A aus (3.1) mit C von rechts führt auf

$$AC\tilde{\boldsymbol{x}} = \boldsymbol{b}, \quad C\tilde{\boldsymbol{x}} = \boldsymbol{x}. \tag{3.25}$$

Man bezeichnet C als *rechten Vorkonditionierer*. Wendet man C von links auf (3.1) an,

$$CA\boldsymbol{x} = C\boldsymbol{b}, \tag{3.26}$$

so wird C als *linker Vorkonditionierer* bezeichnet. Eine dritte Möglichkeit zur Vorkonditionierung eines Gleichungssystems besteht in einem Vorkonditionierer in Produktform,

$$C = C_L C_R, \tag{3.27}$$

wobei C_L, C_R typischerweise Dreiecksmatrizen sind. Die Anwendung führt auf

$$C_L A C_R \tilde{\boldsymbol{x}} = C_L \boldsymbol{b}, \quad C_R \tilde{\boldsymbol{x}} = \boldsymbol{x}.$$

Offensichtlich ist $C = A^{-1}$ ein idealer Vorkonditionierer. Jedoch ist die Berechnung der Inversen in der Regel aufwändiger als die Lösung von (3.1). Zudem ist die Inverse einer schwach besetzten Matrix im Allgemeinen nicht mehr schwach besetzt, d.h. die Besetzungsstruktur der Inversen enthält wesentlich mehr Einträge als die Besetzungsstruktur von A (fill-in). Dies führte zur Entwicklung von Verfahren, welche eine Näherung an die Inverse berechnen und die Besetzungsstruktur von A nahezu erhalten, den sogenannten SPAI (sparse approximate inverses) [15].

Die Multiplikation mit der Inversen kann durch die LU-Faktorisierung mit anschließendem Vorwärts-/Rückwärtslösen simuliert werden:

$$A = LU, \quad \boldsymbol{x} = A^{-1}\boldsymbol{b} \quad \Leftrightarrow \quad L\boldsymbol{z} = \boldsymbol{b} \quad \text{und} \quad U\boldsymbol{x} = \boldsymbol{z}.$$

Da die LU-Faktorisierung aufwändig ist und während des Gaußschen Eliminationsprozesses im Allgemeinen sukzessive neue Nicht-Null-Einträge im Besetzungsmuster generiert werden, sind Verfahren zur approximativen Berechnung der Faktoren L und U entwickelt worden, die die Besetzungsstruktur teilweise erhalten, die sogenannten *unvollständigen Dreieckszerlegungen* [58, 36]. Neben diesen können auch die Algebraischen Mehrgitterverfahren zur Konstruktion von Vorkonditionierern eingesetzt werden [3, 4]. Eine Übersicht über Vorkonditionierungstechniken gibt Benzi in [14].

In dieser Arbeit wird eine neue Möglichkeit zur approximativen Berechnung der Faktoren L und U präsentiert, welche die Technik der hierarchischer Matrizen auf rein algebraische Weise nutzt.

3.7 Das vorkonditionierte Konjugierte-Gradienten-Verfahren

Sei A eine symmetrisch positiv definite Matrix und $M \approx A$ eine Näherung, die symmetrisch positiv definit ist. Nutzen wir in (3.25) oder (3.26) den Vorkonditionierer $C = M^{-1}$, so sind die Matrizen AC bzw. CA im Allgemeinen nicht symmetrisch positiv definit. Somit ist Alg. 5 nicht anwendbar.

Die Matrix AC ist selbstadjungiert bzgl. des C-Skalarprodukts

$$\langle \boldsymbol{x}|\boldsymbol{y}\rangle_C := \langle C\boldsymbol{x}|\boldsymbol{y}\rangle = \langle \boldsymbol{x}|C\boldsymbol{y}\rangle\,,$$

da

$$\langle AC\boldsymbol{x}|\boldsymbol{y}\rangle_C = \langle CAC\boldsymbol{x}|\boldsymbol{y}\rangle = \langle AC\boldsymbol{x}|C\boldsymbol{y}\rangle = \langle C\boldsymbol{x}|AC\boldsymbol{y}\rangle = \langle \boldsymbol{x}|AC\boldsymbol{y}\rangle_C$$

gilt.

Algorithmus 6 PCG-Verfahren mit rechtem Vorkonditionierer

1: $i = 0$
2: $\boldsymbol{r}_i = \boldsymbol{b} - A\boldsymbol{x}_i$, $\boldsymbol{d}_i = \boldsymbol{v}_i = C\boldsymbol{r}_i$, $\varrho_i = \langle \boldsymbol{v}_i|\boldsymbol{r}_i\rangle$
3: **while** $\|\boldsymbol{r}_i\| > \varepsilon$ und $i < i_{\max}$ **do**
4: $\quad \boldsymbol{w}_i = A\boldsymbol{d}_i$
5: $\quad \alpha_i = \dfrac{\varrho_i}{\langle \boldsymbol{d}_i|\boldsymbol{w}_i\rangle}$
6: $\quad \boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{d}_i$
7: $\quad \boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i \boldsymbol{w}_i$
8: $\quad \boldsymbol{v}_{i+1} = C\boldsymbol{r}_{i+1}$
9: $\quad \varrho_{i+1} = \langle \boldsymbol{v}_{i+1}|\boldsymbol{r}_{i+1}\rangle$
10: $\quad \beta_{i+1} = \dfrac{\varrho_{i+1}}{\varrho_i}$
11: $\quad \boldsymbol{d}_{i+1} = \boldsymbol{v}_{i+1} + \beta_{i+1}\boldsymbol{d}_i$
12: $\quad i = i + 1$

Satz 3.5. *Das PCG-Verfahren konvergiert mit der Konvergenzabschätzung*

$$\|\boldsymbol{x}_{i+1} - \boldsymbol{x}\|_A \leq \frac{2q^i}{1+q^{2i}} \|\boldsymbol{x}_0 - \boldsymbol{x}\|_A , \quad q = \frac{\sqrt{\kappa_2(AC)} - 1}{\sqrt{\kappa_2(AC)} + 1}. \tag{3.28}$$

Wird A durch eine approximative Cholesky-Zerlegung faktorisiert so ist $L^{-1}AL^{-T}$ positiv definit. Im PCG-Verfahren für den Vorkonditionierer (3.27) kann das euklidische Skalarprodukt, anstatt dem C-Skalarprodukt wie im obigen Algorithmus, genutzt werden. Lediglich bei der Berechnung des Residuums bzw. der Suchrichtung müssen die Matrizen L^{-1} bzw. L^{-T} berücksichtigt werden.

4 Hierarchische Matrizen

Hierarchische Matrizen ($\mathcal{H}$-Matrizen) wurden von Hackbusch [38], Hackbusch und Khoromskij [41, 40] eingeführt. Mit Hilfe dieser hierarchischen Technik kann eine Näherung einer vollbesetzten Matrix datenschwach[1] berechnet werden. Jedoch lässt sich nicht jede beliebige Matrix als $\mathcal{H}$-Matrix darstellen. Schwach besetzte Matrizen können exakt in das hierarchische Format überführt werden. Die Inverse einer schwach besetzten Matrix ist im Allgemeinen nicht mehr schwach besetzt. In [11, 6, 7] wird die Existenz einer Approximation an die Inverse einer FE-Matrix im hierarchischen Format bewiesen. Die $\mathcal{H}$-Inverse ist dann in linear-logarithmischer Komplexität berechenbar und eignet sich daher als Vorkonditionierer. Die Wirkung der Inversen kann durch die LU-Zerlegung mit Vorwärts-/Rückwärtssubstitution simuliert werden. Zudem ist die $\mathcal{H}$-LU-Zerlegung in der gleichen Zeitkomplexität wie die $\mathcal{H}$-Inverse berechenbar. Der Vergleich zwischen der Implementierung beider Operationen ergibt in der Praxis einen Vorteil der hierarchischen LU-Zerlegung. In [8] wird nachgewiesen, dass die Faktoren der LU-Zerlegung einer FE-Matrix ebenfalls im hierarchischen Format darstellbar sind. Neben der Effizienz sequentieller Verfahren bietet sich zur Beschleunigung das Verteilen der Arbeit auf mehrere Prozessoren an. Dazu wurde in [49, 50] die Parallelisierbarkeit der Algorithmen untersucht.

Seien I, J Indexmengen und A eine $I \times J$-Matrix. Die Konstruktion hierarchischer Matrizen basiert auf zwei Grundbausteinen:

1. auf der Approximation von Unterblöcken $A|_{t_1 \times t_2}$ durch die im Abschnitt 4.1 eingeführten Niedrigrang-Matrizen, wobei $t_1 \times t_2 \subset I \times J$ ist und

2. einer im Abschnitt 4.2 behandelten Partitionierung der Indexmenge $I \times J$.

Im dritten Abschnitt des Kapitels wird die $\mathcal{H}$-Matrix-Arithmetik kurz skizziert, der Speicherverbrauch hierarchischer Matrizen analysiert und die Komplexität wichtiger Matrixoperationen angegeben.

4.1 Niedrigrang-Matrizen

4.1.1 Speicheraufwand

Setzen wir keine strukturellen Eigenschaften für die Matrixeinträge voraus, so wird die Matrix elementweise im Speicher abgelegt. Wir bezeichnen dieses Format als *volles For-*

[1] Eine $n \times n$-Matrix heißt datenschwach, falls die Matrixeinträge mit weniger als n^2 Informationen beschrieben werden können.

mat. Eine alternative Darstellung ist die Form

$$A = UV^H, \tag{4.1}$$

wobei U eine $m \times r$-Matrix und V eine $n \times r$-Matrix sind und die Matrizen jeweils elementweise gespeichert werden. Seien $\boldsymbol{u}_i$, $i = 1, \ldots, r$, die Spaltenvektoren der Matrix U und entsprechend $\boldsymbol{v}_j$, $j = 1, \ldots, r$, die Spaltenvektoren der Matrix V. Dann können wir A in der Form

$$A = \sum_{i=1}^{r} \boldsymbol{u}_i \boldsymbol{v}_i^H$$

schreiben.

Während das volle Format $m \cdot n$ Speichereinheiten benötigt, müssen in der Darstellung (4.1) $m \cdot r + n \cdot r = (m + n) \cdot r$ Speichereinheiten zur Verfügung gestellt werden.

4.1.2 Matrix-Vektor-Multiplikation

Das Produkt einer $m \times n$-Matrix A im vollen Format mit einem Vektor $\boldsymbol{x}$ der Länge n kann mit Alg. 7 berechnet werden. Dazu sind $m \cdot (2n - 1)$ arithmetische Operationen notwendig.

Algorithmus 7 MVM (m, n, A, $\boldsymbol{x}$, $\boldsymbol{y}$)

```
1: for i = 0 : m − 1 do
2:     y(i) = A(0, i) · x(0)
3:     for j = 1 : n − 1 do
4:         y(i) = y(i) + A(i, j) · x(j)
```

Liegt A im Format (4.1) vor, kann die MVM wie in Alg. 8 in zwei Teilaufgaben zerlegt werden. Dieser Algorithmus benötigt $r \cdot (2n - 1) + m \cdot (2r - 1) = 2r(m + n) - r - m$ Operationen.

Algorithmus 8 MVM für Format (4.1) (m, n, r, U, V^H, $\boldsymbol{x}$, $\boldsymbol{y}$)

```
1: MVM (r, n, V^H, x, z)
2: MVM (m, r, U, z, y)
```

4.1.3 Definition Niedrigrang-Matrix

Es sei vorausgesetzt, dass der Rang r der $m \times n$-Matrix A die Bedingung

$$r \cdot (m + n) < m \cdot n \tag{4.2}$$

erfüllt und die Matrix im Format (4.1) vorliegt. Im Vergleich zum vollen Format ist dann sowohl der Speicherverbrauch als auch der Aufwand der MVM geringer. Dies ist Motivation für die Definition der Menge der *Niedrigrang-Matrizen*

$$\mathbb{C}_r^{m \times n} := \{A \in \mathbb{C}^{m \times n} : \operatorname{rank} A \leq r\}.$$

4.1.4 Addition und Multiplikation von Niedrigrang-Matrizen

4.1.4.1 Addition

Seien $A = U_A V_A^H \in \mathbb{C}_{r_A}^{m\times n}$ und $B = U_B V_B^H \in \mathbb{C}_{r_B}^{m\times n}$ gegeben. Die Summe

$$A + B = U_A V_A^H + U_B V_B^H = \begin{bmatrix} U_A & U_B \end{bmatrix} \cdot \begin{bmatrix} V_A^H \\ V_B^H \end{bmatrix}$$

lässt sich ohne arithmetische Operationen nur durch Umorganisation des Speichers realisieren. Allerdings gilt

$$\operatorname{rank}(A + B) \leq \operatorname{rank}(A) + \operatorname{rank}(B) =: r_{A+B},$$

d.h. der Rang der Summe kann auf $r_{A+B} = r_A + r_B$ anwachsen. Die Menge der Niedrigrang-Matrizen ist nicht abgeschlossen bzgl. der Addition. Im Abschnitt 4.1.6 wird eine approximative Addition vorgestellt, die den Rang durch Approximation erhält.

4.1.4.2 Multiplikation

Seien $A = U_A V_A^H \in \mathbb{C}_{r_A}^{m\times p}$ und $B = U_B V_B^H \in \mathbb{C}_{r_B}^{p\times n}$ gegeben, wobei $U_A \in \mathbb{C}^{m\times r_A}, V_A \in \mathbb{C}^{p\times r_A}$ und $U_B \in \mathbb{C}^{p\times r_B}$, $V_B \in \mathbb{C}^{n\times r_B}$. Für das Produkt $AB = U_A V_A^H \cdot U_B V_B^H = UV^H$ gibt es zwei Möglichkeiten der Berechnung:

$$U = U_A V_A^H U_B, \qquad V = V_B, \tag{4.3a}$$

$$U = U_A, \qquad V = V_B (U_B^H V_A). \tag{4.3b}$$

Während der Algorithmus (4.3a) $r_A r_B(2p-1) + m r_B(2r_A - 1)$ Operationen benötigt, sind für Algorithmus (4.3b) $r_A n(2r_B - 1) + r_A r_B(2p - 1)$ Operationen notwendig. In Abhängigkeit der Dimensionen der Eingabematrizen kann der Algorithmus mit der geringeren Anzahl von Operationen ausgewählt werden.

4.1.5 Approximation durch Niedrigrang-Matrizen, Normen

Für den Rang r einer $m \times n$-Matrix A gilt: $r \leq \min\{m, n\}$. Im Allgemeinen ist $r \sim \min\{m, n\}$. In diesem Abschnitt soll mit Hilfe der Singulärwertzerlegung eine Näherung $\tilde{A}$ an A mit kleinerem Rang berechnet und gleichzeitig der Approximationsfehler angeben werden.

4.1.5.1 Singulärwertzerlegung

Satz 4.1. *Sei $A \in \mathbb{R}^{m\times n}$, $m \geq n$. Dann existieren orthogonale Matrizen*

$$U = \begin{bmatrix} \boldsymbol{u}_1, \ldots, \boldsymbol{u}_n \end{bmatrix} \in \mathbb{R}^{m\times n} \quad \text{und} \quad V = \begin{bmatrix} \boldsymbol{v}_1, \ldots, \boldsymbol{v}_n \end{bmatrix} \in \mathbb{R}^{n\times n},$$

so dass

$$U^T A V = \Sigma \text{ bzw. } A = U \Sigma V^T \quad \text{mit} \quad \Sigma = \operatorname{diag}(\sigma_1, \ldots, \sigma_n),$$

wobei für die Singulärwerte $\sigma_1 \geq \cdots \geq \sigma_n$ gilt.

Beweis. siehe [32, Seite 70] □

Die Zerlegung der Matrix $A = U\Sigma V^T$ nennt man Singulärwertzerlegung (engl. singular value decomposition - SVD). Die SVD ist in etwa $21mn^2$ Operationen zu berechnen, siehe [32, Abschnitt 5.4.5]. Daher wird die SVD im Allgemeinen nur auf relativ kleine Matrizen angewendet.

Bei der näherungsweisen Darstellung der Matrix A durch

$$\tilde{A} := U\Sigma_r V^H, \quad \Sigma_r := \operatorname{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0)$$

entsteht ein Fehler. Der Abstand zwischen A und $\tilde{A}$, also die Approximationsqualität, wird durch eine Norm gemessen.

4.1.5.2 Die Frobeniusnorm

Die Frobeniusnorm einer $m \times n$-Matrix A ist wie folgt definiert

$$\|A\|_F := \left(\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2 \right)^{\frac{1}{2}} = \sqrt{\operatorname{trace}(A^H A)}. \tag{4.4}$$

Bei elementweiser Speicherung von A beträgt der Aufwand zur Berechnung der Frobeniusnorm $n \cdot m$ Operationen. Im Format (4.1) ergibt sich die Berechnungsmöglichkeit

$$\|UV^H\|_F^2 = \sum_{k,\ell=1}^{r} \left(\boldsymbol{u}_k^H \boldsymbol{u}_\ell \right) \left(\boldsymbol{v}_k \boldsymbol{v}_\ell^H \right) \tag{4.5}$$

mit einem Aufwand von $2r^2(m+n)$. Die Darstellung (4.1) ist nur vorteilhaft, solange die im Vergleich zu (4.2) schärfere Bedingung $r^2(m+n) < m \cdot n$ gilt.

4.1.5.3 Die Spektralnorm

Die Spektralnorm einer $m \times n$-Matrix A ist

$$\|A\|_2 := \max_{\|\boldsymbol{x}\|_2 = 1} \|A\boldsymbol{x}\|_2 = \sqrt{\lambda_{\max}(A^H A)}.$$

Die Menge aller Eigenwerte einer $n \times n$-Matrix M bezeichnet man als *Spektrum*:

$$\sigma(M) = \{\lambda_1, \ldots, \lambda_n\}.$$

Wegen $\sigma(AB)\backslash\{0\} = \sigma(BA)\backslash\{0\}$ folgt mit $A = V$ und $B = U^H U V^H$ die Gleichung

$$\|A\|_2 = \sqrt{\lambda_{\max}\left((UV^H)^H (UV^H) \right)} = \sqrt{\lambda_{\max}\left((U^H U)(V^H V) \right)}.$$

Der Aufwand zur Berechnung der Spektralnorm von A reduziert sich auf den Aufwand für die Berechnung des größten Eigenwerts der $r \times r$-Matrix $U^H U V^H V$.

Satz 4.2. *Sei die SVD von* $A = U\Sigma V^T$ *gegeben, wobei* $A \in \mathbb{R}^{m\times n}$. *Dann gilt*

$$\min_{\tilde{A}\in\mathbb{C}_r^{m\times n}} \|A - \tilde{A}\|_2 = \|\Sigma - \Sigma_r\|_2,$$

wobei $\Sigma_r = \operatorname{diag}(\sigma_1, \ldots, \sigma_r, 0, \ldots, 0)$ *und* $\tilde{A} = A_r = U\Sigma_r V^T$.

Der in der Spektralnorm gemessene Approximationsfehler ist

$$\|A - A_r\|_2 = \|\Sigma - \Sigma_r\|_2 = \sigma_{r+1}. \tag{4.6}$$

Beweis. siehe [53] □

In der Frobeniusnorm ergibt sich der Approximationsfehler

$$\|A - A_r\|_F = \sqrt{\sum_{\ell=r+1}^{n} \sigma_\ell^2}, \tag{4.7}$$

vorausgesetzt $m \geq n$.

Zu vorgegebener Genauigkeit ε kann entsprechend der Rang r berechnet werden. Umgekehrt kann bei vorgegebenen Rang der Approximationsfehler durch (4.6) oder (4.7) angegeben werden. Wie wir in den folgenden Abschnitten sehen werden, approximieren wir nicht nur Matrizen mit nahezu vollem Rang, sondern wenden die SVD auf Niedrigrang-Matrizen an. Der Aufwand wird durch Alg. 9 im Vergleich zur SVD, angewendet auf das volle Matrixformat, erheblich reduziert.

Algorithmus 9 SVD für die Niedrigrang-Matrixrepräsentation (4.1)

1: berechne die QR-Zerlegungen: $U = Q_U R_U$ und $V = Q_V R_V$
2: berechne die $r \times r$-Matrix $T = R_U R_V^H$
3: berechne die SVD der $r \times r$-Matrix $T = \hat{U}\hat{\Sigma}\hat{V}^H$
4: berechne $Q_U\hat{U}$ und $(Q_V\hat{V})^H \quad \rightarrow \quad A = (Q_U\hat{U})\,\hat{\Sigma}\,(Q_V\hat{V})^H$

Für die mit dem Algorithmus von Householder realisierten QR-Zerlegungen im Schritt Eins werden $2r^2(m+n)$ Operationen benötigt, wobei statt Q_U bzw. Q_R jeweils nur die Householder-Reflektoren gespeichert werden. Die Matrix-Matrix-Multiplikation im Schritt Zwei lässt sich unter Ausnutzung der Struktur der Einträge in R_U und R_V in der Komplexität $\frac{2}{3}r^3 + \frac{1}{3}r$ berechnen. Die SVD der $r \times r$-Matrix $R_U R_V^H$ ist in einem Aufwand von $21r^3$ durchzuführen. Zur Berechnung der unitären Matrizen $Q_U\hat{U}$ und $Q_V\hat{V}$ im letzten Schritt müssen die im ersten Schritt berechneten Householder-Reflektoren auf die Matrizen $\hat{U}$ und $\hat{V}$ angewendet werden. Dazu benötigt man $4r^2m + 4r^2n = 4r^2(m+n)$ Operationen. Die Gesamtzahl der Operationen ist $\mathcal{W}_{\text{Alg. 9}} = \frac{65}{3}r^3 + 6r^2(m+n)$.

4.1.6 Approximative Addition von Niedrigrang-Matrizen

Der anwachsende Rang bei der (exakten) Addition führt zu einer größeren Speicherbelegung und längeren Laufzeiten der numerischen Algorithmen. Deshalb ist die Definition einer näherungsweisen Addition wünschenswert, deren Ergebnis wieder in $\mathbb{C}_r^{m\times n}$ liegt.

Die Anwendung des Alg. 9 auf $U = \begin{bmatrix} U_A & U_B \end{bmatrix}$ und $V = \begin{bmatrix} V_A & V_B \end{bmatrix}$ liefert eine Approximation an das exakte Ergebnis. Man bezeichnet diese Operation als *gerundete Addition.*

Satz 4.3. *Seien $\|\cdot\|$ eine unitär invariante Norm, $A \in \mathbb{C}_{r_A}^{m\times n}$ und $B \in \mathbb{C}_{r_B}^{m\times n}$. Man kann die Matrix $S \in \mathbb{C}_r^{m\times n}$, $r \le r_A + r_B$, mit*

$$\|A + B - S\| = \inf_{M\in\mathbb{C}_r^{m\times n}} \|A + B - M\| \tag{4.8}$$

in $\mathcal{O}\left(\frac{65}{3}(r_A + r_B)^3 + 6(r_A + r_B)^2(m + n)\right)$ Operationen berechnen.

Beweis. Die Existenz einer (4.8) erfüllenden Matrix S folgt aus Satz 4.2. Mit Alg. 9 lässt sich S in der angegebenen Anzahl von Operationen berechnen. □

4.1.7 Agglomeration von Niedrigrang-Matrizen

Das Ziel der Agglomeration von Niedrigrang-Matrix-Blöcken zu einem neuen Niedrigrang-Matrix-Block besteht in der Reduktion des Speicherverbrauchs. Sei

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & \end{bmatrix} + \begin{bmatrix} & A_{12} \end{bmatrix} + \begin{bmatrix} & \\ A_{21} & \end{bmatrix} + \begin{bmatrix} & \\ & A_{22} \end{bmatrix} \tag{4.9}$$

wobei A_{11} eine $m_1 \times n_1$, A_{12} eine $m_1 \times n_2$, A_{21} eine $m_2 \times n_1$ und A_{22} eine $m_2 \times n_2$ Matrizen sind. Fasst man die Blöcke durch die im Abschnitt 4.1.6 definierte gerundete Addition zusammenfassen, dann erfordert die Berechnung der Faktoren U und V für die Form (4.1) mit Alg. 9 einen Aufwand von

$$\mathcal{O}\left(\frac{65}{3}\left(r_{11} + r_{12} + r_{21} + r_{22}\right)^3 + 6\left(r_{11} + r_{12} + r_{21} + r_{22}\right)^2 (m + n)\right)$$

bzw. mit $r := \max\{r_{11}, r_{12}, r_{21}, r_{22}\}$ den Aufwand $\mathcal{O}\left(1387r^3 + 96r^2(m + n)\right)$.

Der folgenden Algorithmus nutzt die spezielle Struktur der Terme auf der rechten Seite von (4.9) aus. Sei

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} U_{11}V_{11}^H & U_{12}V_{12}^H \\ U_{21}V_{21}^H & U_{22}V_{22}^H \end{bmatrix} = \begin{bmatrix} U_{11} & U_{12} & & \\ & & U_{21} & U_{22} \end{bmatrix} \cdot \begin{bmatrix} V_{11}^H & \\ & V_{12}^H \\ V_{21}^H & \\ & V_{22}^H \end{bmatrix}$$

mit $U_{11} \in \mathbb{R}^{m_1\times r}, V_{11} \in \mathbb{R}^{n_1\times r}$, $U_{12} \in \mathbb{R}^{m_1\times r}, V_{12} \in \mathbb{R}^{n_2\times r}$, $U_{21} \in \mathbb{R}^{m_2\times r}, V_{21} \in \mathbb{R}^{n_1\times r}$ und $U_{22} \in \mathbb{R}^{m_2\times r}, V_{22} \in \mathbb{R}^{n_2\times r}$. Im ersten Schritt werden die vier QR-Zerlegungen

$$\begin{aligned} [U_{11}, U_{12}] &= Q_{U_{11}U_{12}} R_{U_{11}U_{12}}, & [U_{21}, U_{22}] &= Q_{U_{21}U_{22}} R_{U_{21}U_{22}}, \\ [V_{11}, V_{21}] &= Q_{V_{11}V_{21}} R_{V_{11}V_{21}}, & [V_{12}, V_{22}] &= Q_{V_{12}V_{22}} R_{V_{12}V_{22}} \end{aligned}$$

durchgeführt, wobei nur die Householder-Reflektoren gespeichert und die unitären Q nicht explizit gebildet werden. Symbolisch ergibt sich

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} Q_{U_{11}U_{12}} & \\ & Q_{U_{21}U_{22}} \end{bmatrix} \begin{bmatrix} R_{U_{11}U_{12}} & \\ & R_{U_{21}U_{22}} \end{bmatrix} \begin{bmatrix} R_{V_{11}V_{21}}^H & \\ & R_{V_{12}V_{22}}^H \end{bmatrix} \begin{bmatrix} Q_{V_{11}V_{21}}^H & \\ & Q_{V_{12}V_{22}}^H \end{bmatrix}.$$

vier QR-Zerlegungen		$2(2r)^2(m+n) = 8r^2(m+n)$
zwei Matrix-Matrix-Multiplikationen		$2 \cdot \left[\frac{2}{3}(2r)^3 + \frac{1}{3}(2r)\right] = \frac{32}{3}r^3 + \frac{4}{3}r$
zwei SVDs		$2 \cdot \left[21(2r)^3\right] = 336r^3$
Anwendung der Householder-Reflektoren		$4(2r)^2(m+n) = 16r^2(m+n)$
Multiplikation der Diagonalmatrix $\bar{\Sigma}$		$r \min\{m,n\}$
	$\approx$	$24r^2(m+n) + 347r^3$

Tabelle 4.1: Anzahl der Operationen für die Agglomeration

Zur Bildung der Blockdiagonalmatrix

$$T = \begin{bmatrix} T_{11} & 0 \\ 0 & T_{22} \end{bmatrix} = \begin{bmatrix} R_{U_{11}U_{12}} & \\ & R_{U_{21}U_{22}} \end{bmatrix} \begin{bmatrix} R^H_{V_{11}V_{21}} & \\ & R^H_{V_{12}V_{22}} \end{bmatrix}$$

werden zwei Matrix-Matrix-Multiplikationen der Dimension $(2r) \times (2r)$ benötigt. Die Berechnung der SVD der $(4r) \times (4r)$-Matrix $T = \bar{U}\bar{\Sigma}\bar{V}^H$ liefert die unitären Faktoren

$$\bar{U} = \begin{bmatrix} \bar{U}_{11} & \\ & \bar{U}_{22} \end{bmatrix} \text{ und } \bar{V} = \begin{bmatrix} \bar{V}^H_{11} & \\ & \bar{V}^H_{22} \end{bmatrix}$$

in Block-Diagonalform, und es ist

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \underbrace{\begin{bmatrix} Q_{U_{11}U_{12}} & \\ & Q_{U_{21}U_{22}} \end{bmatrix}}_{Q_U} \begin{bmatrix} \bar{U}_{11} & \\ & \bar{U}_{22} \end{bmatrix} \begin{bmatrix} \bar{\Sigma}_{11} & \\ & \bar{\Sigma}_{22} \end{bmatrix} \begin{bmatrix} \bar{V}^H_{11} & \\ & \bar{V}^H_{22} \end{bmatrix} \underbrace{\begin{bmatrix} Q_{V^H_{11}V^H_{21}} & \\ & Q_{V^H_{12}V^H_{22}} \end{bmatrix}}_{Q_V}.$$

Die Householder-Reflektoren $Q_{U_{11}U_{12}}$, $Q_{U_{21}U_{22}}$, $Q^H_{V_{11}V_{22}}$ sowie $Q^H_{V_{21}V_{22}}$ können blockweise auf $\bar{U}_{11}$, $\bar{U}_{22}$, $\bar{V}^H_{11}$ sowie $\bar{V}^H_{22}$ angewendet werden. Durch die Multiplikation der Diagonalmatrix $\bar{\Sigma}$ entweder mit der Block-Diagonalmatrix $Q_U\bar{U}$ oder der Block-Diagonalmatrix $\bar{V}^H Q_V$ erhält man die Faktoren der Niedrigrang-Darstellung der Matrix A.

In der Tabelle 4.1 werden die Kosten pro Schritt aufgeschlüsselt. Zusammengefasst werden zur Agglomeration $\mathcal{O}\left(24r^2(m+n) + 347r^3\right)$ Operationen nötig.

4.2 Partitionierung der Matrix in Blöcke

Da nicht jede Matrix $A \in \mathbb{R}^{I \times J}$ direkt als Niedrigrang-Matrix darstellbar bzw. als Niedrigrang-Matrix approximierbar ist, suchen wir nach einer Partitionierung der Matrix in Blöcke, welche als Niedrigrang-Matrix approximierbar oder klein sind. Es ist sowohl für die Speichereffizienz als auch für die angestrebte linear-logarithmische Komplexität wichtiger Matrix-Operationen entscheidend, eine geeignete Partition der Indexmenge $I \times J$ in linear-logarithmischer Zeit zu finden.

Definition 4.4. Seien I, J endliche Indexmengen. Eine Teilmenge P der Potenzmenge $\mathcal{P}(I \times J)$ nennt man *Matrixpartition* (*Partition*), wenn

$$I \times J = \bigcup_{b \in P} b \quad \text{und} \quad b_i \cap b_j = \emptyset, \quad i \neq j,$$

gilt. Ein Element $b \in P$ bezeichnet man dann als *Indexblock* (*Block*).

Um möglichst wenig Speicherplatz zu belegen, sollte eine geeignete Partition wenige Blöcke haben. Andererseits ist es wünschenswert, dass die Partition ausreichend viele Blöcke enthält, um jeden Block mit niedrigem Rang zu repräsentieren, d.h. der Rang des Blocks genügt der Bedingung (4.2). Diese beiden Anforderungen an eine geeignete Partition sind diametral, so dass eine Gewichtung der Anforderungen vorgenommen werden muss. Eine dritte Bedingung ist an die Anordnung der Blöcke zu stellen – die Blockstruktur soll effiziente Matrixoperationen erlauben.

Die Anzahl möglicher Permutationen von Zeilen und Spalten beträgt $|I|! \cdot |J|!$. Eine einfache Suche durch alle Partitionen ist daher zu aufwändig. Tensorpartitionen besitzen eine mehr als linear-logarithmische Komplexität bzgl. des Speicherverbrauchs, die Matrixoperationen sind ebenfalls nicht in linear-logarithmischer Zeit durchzuführen, siehe [9].

Zur Suche geeigneter Partitionen hat sich ein hierarchischer Zugang bewährt. Dazu wird zunächst ein Clusterbaum berechnet, der die hierarchische Partition der Indexmenge I speichert. Auf dem Clusterbaum basierend wird ein sogenannter Block-Clusterbaum konstruiert, dessen Blätter eine zulässige Partition enthalten.

In Abschnitt 4.2.1 wird eine für Finite-Elemente-Anwendungen typische Bedingung eingeführt, die ein zulässiger Block erfüllen muss. Unter Berücksichtigung dieser Zulässigkeitsbedingung werden in Abschnitt 4.2.2 Konstruktionen des Clusterbaums angegeben.

Auf Basis des Clusterbaums wird in Abschnitt 4.2.3 der Aufbau des Block-Clusterbaums erklärt, wofür eine *effiziente* Auswertung der Zulässigkeitsbedingung benötigt wird. Die gesuchte Matrixpartition ist in den Blättern des Block-Clusterbaums enthalten.

4.2.1 Zulässigkeitskriterium

Das Zulässigkeitskriterium soll folgende Eigenschaften haben.

1. Ist $b \in P$ zulässig, dann fallen die Singulärwerte von $A|_b$ exponentiell ab.
2. Ist $b \in P$ zulässig, dann sind die Unterblöcke $b' \subset b$ ebenfalls zulässig.
3. Das Kriterium kann für einen Block $b = t \times s \subset I \times J$ mit einem numerischen Aufwand von $\mathcal{O}(|t| + |s|)$ überprüft werden.

Definition 4.5. Eine Partition P heißt *zulässig*, wenn die Blöcke $b = t_1 \times t_2 \in P$ zulässig oder klein sind. Ein Block wird als klein bezeichnet, wenn $|t_1| \leq n_{\min}$ und $|t_2| \leq n_{\min}$ gilt.

Die Konstante $n_{\min}$ wird abhängig von der Rechnerarchitektur und insbesondere der Cache-Größe gewählt.

4.2.1.1 Geometrisches Zulässigkeitskriterium für FE-Matrizen

Der zu (1.3) inverse Differentialoperator kann als Integraloperator

$$(\mathcal{L}^{-1}\,\varphi)(x) = \int_\Omega G(x,y)\varphi(y)\mathrm{d}y \tag{4.10}$$

dargestellt werden. Seien $X_i = \operatorname{supp}\varphi_i$ und $X_{t_1} = \bigcup_{i\in t_1} X_i$ bzw. $X_{t_2} = \bigcup_{i\in t_2} X_i$ disjunkte Teilgebiete von Ω. Die *Greensche Funktion* $G(x,y)$ ist für $x = y$ singulär. Unter der Bedingung, dass die Teilgebiete X_{t_1}, X_{t_2} genügend großen Abstand haben, kann für Differentialoperatoren mit hinreichend glatten Koeffizienten und hinreichend glatten Rand $\partial\Omega$ gezeigt werden, dass $G(x,y)$ auf $X_{t_1}\times X_{t_2}$, durch eine *separable Entwicklung* darstellbar ist, d.h.

$$G(x,y) = \sum_{\ell=1}^{r} p_\ell(x)q_\ell(y) + R_r(x,y), \quad x\in X_{t_1}, y\in X_{t_2}, \tag{4.11}$$

wobei p_ℓ bzw. q_ℓ hier Polynome in x bzw. y sind. Der Zahl r der Summanden in $\sum_{\ell=1}^{r} p_\ell(x)q_\ell(y)$ heißt *Separationsrang*. Eine separable Entwicklung heißt *exponentiell konvergent*, falls Konstanten $c_1, c_2, \alpha \geq 0$ existieren, so dass

$$\|R_r\| \leq c_1 \exp(-c_2 r^\alpha)$$

ist. Unter den obigen Voraussetzungen ist die separable Entwicklung (4.11) exponentiell konvergent. Die Anwendung der Galerkin-Diskretisierung mit den FE-Basisfunktionen φ_i führt auf die Matrix B mit Einträgen

$$b_{\nu\mu} := \int_\Omega\int_\Omega \varphi_\nu(x)G(x,y)\varphi_\mu(y)\mathrm{d}x\mathrm{d}y. \tag{4.12}$$

Seien $I_{X_{t_1}} = \{\nu : \operatorname{supp}\varphi_\nu \subset \overline{X_{t_1}}\}$ und $I_{X_{t_2}} = \{\mu : \operatorname{supp}\varphi_\mu \subset \overline{X_{t_2}}\}$. Setzen wir $G(x,y) \approx \sum_{\ell=1}^{r} p_\ell(x)q_\ell(y)$ in (4.12) ein, so erhalten wir

$$b_{\nu\mu} \approx \tilde{b}_{\nu\mu} := \sum_{\ell=1}^{r} \int_\Omega \varphi_\nu(x)p_\ell(x)\mathrm{d}x \cdot \int_\Omega \varphi_\mu(y)q_\ell(y)\mathrm{d}y.$$

Der Block $B|_{X_{t_1}\times X_{t_2}}$ hat daher höchstens Rang r. Dies motiviert eine *geometrische Zulässigkeitbedingung*

$$\min\{\operatorname{diam} X_{t_1}, \operatorname{diam} X_{t_2}\} \leq \eta \operatorname{dist}(X_{t_1}, X_{t_2}). \tag{4.13}$$

Bemerkung 4.6.

1. Die Existenz einer Approximation an die Inverse der FE-Matrix elliptischer Operatoren kann sogar für L^∞-Koeffizienten gezeigt werden [11].

2. Das Hauptaugenmerk dieser Arbeit soll auf die Vorkonditionierung iterativer Löser für schwach besetzte Gleichungssysteme, wie etwa FE-Systeme, gerichtet sein. Da die Finite-Elemente-Matrizen in unseren Anwendungsfällen quadratisch sind, werden wir uns auf die Partition von $I\times I$ statt auf $I\times J$ konzentrieren.

4.2.2 Clusterbaum

Definition 4.7. Man nennt den Graphen $T_I = (V_{T_I}, E_{T_I})$ mit den Knoten V_{T_I} und den Kanten E_{T_I} *Clusterbaum* zur Indexmenge I, falls

1. die Indexmenge I die Wurzel des Baumes bildet,
2. falls die Sohnknoten paarweise disjunkte Indexmengen enthalten, d.h. $t = \cup_{t' \in S_I(t)} t'$ (für alle $t \in V_{T_I} \backslash \mathcal{L}(T_I)$) und für $t', t'' \in S_I(t) : \ t' \cap t'' = \emptyset$ und
3. alle Knoten, mit Ausnahme der Blätter, mindestens zwei Söhne besitzen.

Der Clusterbaum enthält in jeder Stufe eine Partition der Indexmenge. Mit zunehmender Tiefe wird die Partition immer feiner.

Lemma 4.8. *Der so definierte Clusterbaum hat eine in $|I|$ lineare Anzahl von Knoten.*

Beweis. siehe [9] □

Um eine linear-logarithmische Laufzeit zur Berechnung des Vorkonditionierers nicht zu überschreiten, wird sich herausstellen, dass der Clusterbaum eine Tiefe logarithmisch in $|I|$ haben sollte.

Definition 4.9. Ein Clusterbaum T_I ist *balanciert*, falls eine von $|I|$ unabhängige Konstante c existiert, so dass

$$\min_{t \in T_I \backslash \mathcal{L}(T_I)} \left\{ \frac{|t_1|}{|t_2|} : t_1, t_2 \in S(t) \right\} \leq c$$

gilt.

Lemma 4.10. *Die Tiefe $L(T_I)$ eines balancierten Clusterbaums ist logarithmisch in $|I|$.*

Beweis. siehe [9] □

Bei der FE-Methode sollen durch die Anwendung der Sohnabbildung S_I auf die Cluster t_1, t_2 die Durchmesser von $t_1', t_1'' \subset t_1$ sowie $t_2', t_2'' \subset t_2$ möglichst minimiert werden. Dann werden die Unterblöcke $t_1' \times t_2'$, $t_1' \times t_2''$, $t_1'' \times t_2'$ und $t_1'' \times t_2''$ die zur Clusterung passende Zulässigkeitsbedingungen (4.13) bereits nach wenigen Teilungsschritten erfüllen, und die Partition wird nicht zu fein. Bisher wird der Clusterbaum unter Nutzung der geometrischen Informationen erstellt. Zur geometrischen Clusterung werden beispielsweise die Bounding-Box-Methode oder die Hauptkomponentenanalyse eingesetzt, die im Folgenden kurz erklärt werden. Für schwach besetzte Matrizen bieten sich weitere Arten der Clusterung an. Diese benötigen keine geometrischen Informationen mehr, da der Matrixgraph die Grundlage der Partitionierung bildet (siehe Kapitel 5).

4.2.2.1 Clusterung in regelmäßige Teilquader

Bei der auch Bounding-Box-Clusterung genannten Methode wird ein minimaler achsenparalleler Quader $Q = \prod_{k=1}^{d} [a_k, b_k]$ berechnet, der das gesamte Gebiet Ω bzw. dessen Diskretisierung $X_I = \cup_{i \in I} X_i$ umfasst, wobei $X_i = \operatorname{supp} \varphi_i$ der in Abschnitt 1.2 eingeführte Träger zur i-ten Basisfunktion ist. Durch die rekursive Teilung, senkrecht zur Richtung der jeweils größten Ausdehnung, entstehen Untergebiete halber Größe. Die zugehörigen $\xi_i \in X_i$ können nun entweder dem Sohn t_1 oder t_2 zugeordnet werden.

Algorithmus 10 BoundingBox-Clusterung (Q_t, t)

bestimme j so dass $\max_{1 \le j \le d} b_j - a_j$
$Q_{t_1} := \prod_{k=1}^{j-1} [a_k, b_k] \times [a_j, \frac{1}{2}(b_j + a_j)] \times \prod_{k=j-1}^{d} [a_k, b_k]$
$Q_{t_2} := \prod_{k=1}^{j-1} [a_k, b_k] \times [\frac{1}{2}(b_j + a_j), b_j] \times \prod_{k=j-1}^{d} [a_k, b_k]$
for all $i \in t$ **do**
    if $(\xi_i)_j \le \frac{1}{2}(b_j + a_j)$ **then**
        $t_1 := t_1 \cup i$
    else
        $t_2 := t_2 \cup i$
if $|t_1| > n_{\min}$ **then**
    BoundingBox-Clusterung (Q_{t_1}, t_1)
if $|t_2| > n_{\min}$ **then**
    BoundingBox-Clusterung (Q_{t_2}, t_2)

Mit Alg. 10 kann ein *geometrisch* balancierter Clusterbaum erstellt werden. Der Algorithmus berücksichtigt aber komplexe Geometrien ungenügend, was auf die Kardinalität bezogen zu einem unbalancierte Clusterbaum führen kann.

Beispiel 4.11. Die Clusterung in regelmäßige Teilquader einer regelmäßigen Diskretisierung der in Abb. 4.1 dargestellten Geometrie führt zu dem in Abb. 4.2 skizzierten *unbalancierten* Clusterbaum.

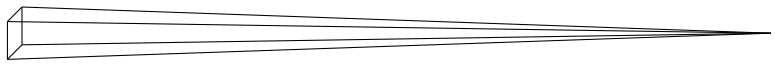


Abb. 4.1: Tetraeder

4.2.2.2 Hauptkomponenten-Clusterung

Die Clusterung nach Hauptkomponenten oder Hauptkomponentenanalyse (engl. principle component analysis, kurz PCA) wird in der Statistik zur Klassifizierung von Elementen einer Stichprobe angewendet. Die durch die Hauptrichtung und das Zentrum

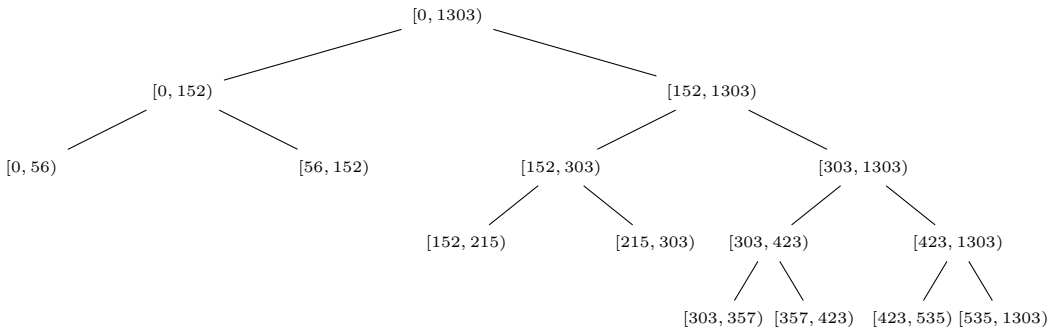


Abb. 4.2: 4 Stufen des durch Zerlegung in regelmäßigen Teilquader generierten Clusterbaums für Abb. 4.1

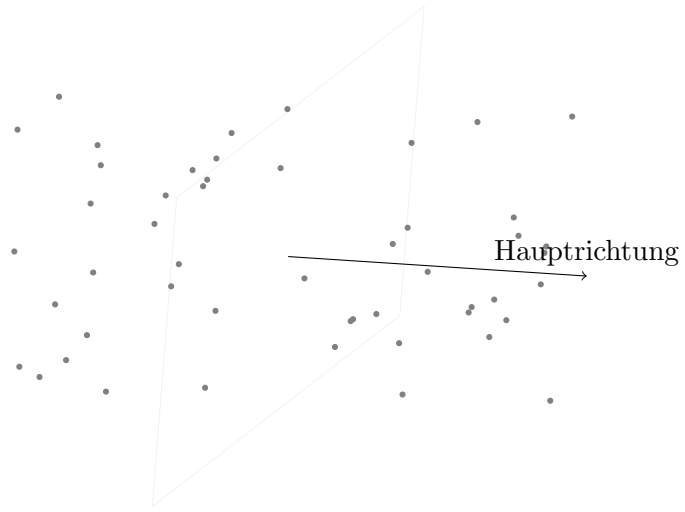


Abb. 4.3: PCA

einer Stichprobe definierte Hyperebene dient als Diskriminanzfunktion. Interpretieren wir die Mittelpunkte $\xi_i \in X_i$, $i \in t$, als Stichprobe und definieren durch $\boldsymbol{s}_t = \frac{1}{|t|} \sum_{i \in t} \xi_i$ das Zentrum des Clusters t, so ist die Hauptrichtung der zum maximalen Eigenwert der symmetrisch, positiv definiten Kovarianzmatrix

$$C_t = \sum_{i \in t} (\xi_i - \boldsymbol{s}_t)(\xi_i - \boldsymbol{s}_t)^T$$

gehörige Eigenvektor $\boldsymbol{u}_t$. Die Teilung des Clusters t kann anhand des Vorzeichens des Skalarprodukts von $\boldsymbol{u}_t$ mit $\xi_i - \boldsymbol{s}_t$ bestimmt werden, etwa durch

$$t_1 := \{i \in t : \boldsymbol{u}_t(\xi_i - \boldsymbol{s}_t) < 0\} \quad \text{und} \quad t_2 = t \backslash t_1.$$

Eine zweite Variante besteht in der Umordnung der Indizes anhand der Größe der Projektion auf $\boldsymbol{u}_t$, d.h. der Vektor $(\boldsymbol{u}_t(\xi_1 - \boldsymbol{s}_t), \ldots, \boldsymbol{u}_t(\xi_n - \boldsymbol{s}_t))$ wird sortiert. Sei π eine entsprechende Permutation, dann ergibt sich $t_1 := \{\pi(1), \ldots, \pi(|t|/2)\}$ und $t_2 := t \backslash t_1$.

Die zweite Variante führt auf kardinalitätsbalancierte Clusterbäume. Der größte Eigenwert und der zugehörige Eigenvektor der $d \times d$-Matrix können durch die Potenzmethode (auch von-Mises-Verfahren) bestimmt werden.

Beispiel 4.12. Die Diskretisierung der in Abb. 4.1 dargestellten Geometrie führt bei Clusterung mit der zweiten Variante zu dem in Abb. 4.4 skizzierten Clusterbaum. Die Knoten der dritten Ebene enthalten bereits unterschiedlich viele Inidzes, was bei einer weiteren Partitionierung auf einen unbalancierten Clusterbaum führt. Insgesamt liefert die PCA-Methode einen nicht so stark unbalancierten Clusterbaum wie Alg. 10.

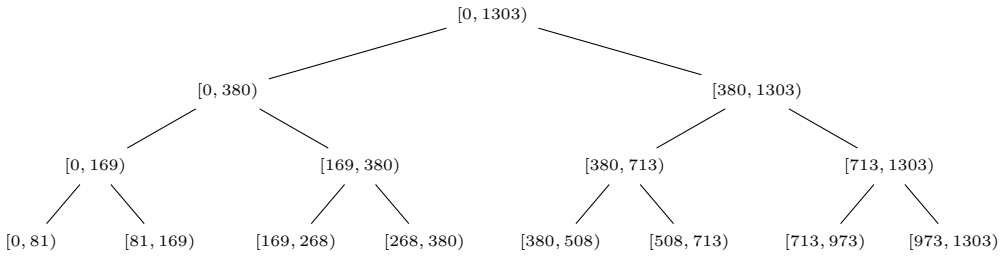


Abb. 4.4: 3 Stufen des durch Zerlegung nach der Hauptkomponentenmethode generierten Clusterbaums für Abb. 4.1

4.2.3 Block-Clusterbaum

4.2.3.1 Definitionen und Theorie

Definition 4.13. Wir nennen einen Graphen $T_{I\times J}$ *Block-Clusterbaum*, wenn

1. $I \times J$ die Wurzel ist,
2. $t_1 \times t_2 = \bigcup_{t_1' \times t_2' \in S_{I\times J}(t_1 \times t_2)} t_1' \times t_2'$ und $(t_1' \times t_2') \cap (t_1'' \times t_2'') = \emptyset$ für $t_1' \times t_2' \neq t_1'' \times t_2''$,
3. $|S_{I\times J}(t_1 \times t_2)| \geq 2$ falls $S_{I\times J}(t_1 \times t_2)$ nicht leer ist.

Die Sohnabbildung $S_{I\times J}$ ist definiert durch

$$S_{I\times J}(t_1 \times t_2) := \begin{cases} \emptyset, & \text{falls } t_1 \times t_2 \text{ zulässig,} \\ S_I(t_1) \times S_J(t_2), & \text{sonst.} \end{cases} \tag{4.14}$$

Der Block-Clusterbaum $T_{I\times J}$ wird durch die rekursive Anwendung von $S_{I\times J}$ auf $I \times J$ konstruiert.

Mit einem Block-Clusterbaum kann die in [33] eingeführte Konstante c_{sp} verknüpft werden. Diese stellt ein Maß für die Komplexität einer Partition dar, und wird in Abschätzungen für Speichernutzung und Laufzeitanalyse von Algorithmen in Abschnitt 4.3 häufig benutzt.

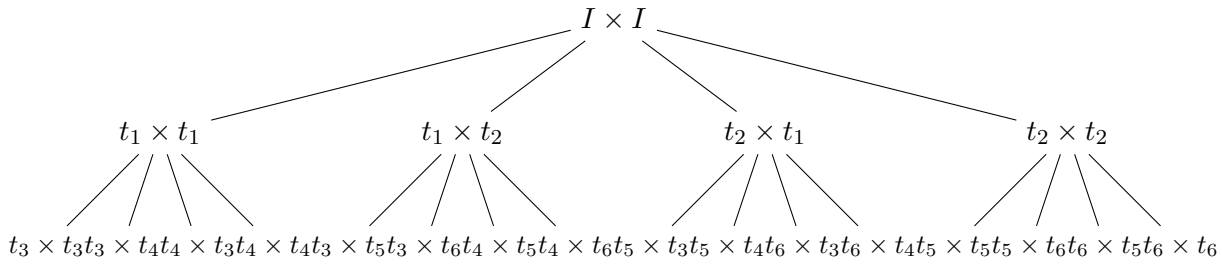


Abb. 4.5: Block-Clusterbaum zu binärem Clusterbaum

Definition 4.14. Seien T_I und T_J Clusterbäume zu den Indexmengen I und J und $T_{I\times J}$ der Block-Clusterbaum für $I \times J$. Man bezeichnet für ein Cluster $t_1 \in T_I$ die maximale Anzahl von Blöcken in $t_1 \times t_2 \in T_{I\times J}$ mit

$$c_{\mathrm{sp}}^{(r)}(T_{I\times J}, t_1) := |\{t_2 \in T_J : t_1 \times t_2 \in T_{I\times J}\}|$$

und für ein Cluster $t_2 \in T_J$ die maximale Anzahl von Blöcken in $t_1 \times t_2 \in T_{I\times J}$ mit

$$c_{\mathrm{sp}}^{(c)}(T_{I\times J}, t_2) := |\{t_1 \in T_I : t_1 \times t_2 \in T_{I\times J}\}|\,.$$

Das Maximum aus diesen beiden Konstanten ist

$$c_{\mathrm{sp}}(T_{I\times J}) := \max\{\max_{t_1\in T_I} c_{\mathrm{sp}}^{(r)}(T_{I\times J}, t_1), \max_{t_2\in T_J} c_{\mathrm{sp}}^{(c)}(T_{I\times J}, t_2)\}. \tag{4.15}$$

Für finite Elemente auf regulären Gittern ist c_{sp} unabhängig von $|I|$ beschränkt [9].

Die Komplexität einer Partition ergibt sich aus der Summe der Kosten pro Block. Im Allgemeinen hängen die Kosten pro Block von dessen Größe ab.

Lemma 4.15. *Unter der Annahme, dass die Kosten des Blocks $t_1 \times t_2$ im Format* (4.1) $r\cdot(|t_1|+|t_2|)$ *und im vollen Format $|t_1|\cdot|t_2|$ betragen, ist die Komplexität einer zulässigen Partition beschränkt durch*

$$\max\{n_{\min}, r\}\ c_{\mathrm{sp}}\ \ (L(T_I)|I| + L(T_J)|J|)\,. \tag{4.16}$$

Beweis. Blöcke $t_1 \times t_2$ der Form (4.1) haben die Komplexität $r(|t_1|+|t_2|)$. Liegen Blöcke $t_1 \times t_2$ im vollen Format vor, so sind diese nach den Anforderungen an eine zulässige Partition klein, d.h. $\min\{|t_1|, |t_2|\} \le n_{\min}$. Zudem gilt

$$|t_1| \cdot |t_2| = \min\{|t_1|, |t_2|\} \max\{|t_1|, |t_2|\} \le n_{\min}(|t_1| + |t_2|). \tag{4.17}$$

Deswegen sind die maximalen Kosten pro Block $\max\{n_{\min}, r\}(|t_1| + |t_2|)$. Mit $c =$

$\max\{n_{\min}, r\}$ ergibt sich für die Gesamtkomplexität einer zulässigen Partition:

$$\begin{aligned}
\mathcal{W} &\leq \sum_{t_1\times t_2\in T_{I\times J}} c(|t_1|+|t_2|) = c\left(\sum_{t_1\times t_2\in T_{I\times J}} |t_1| + \sum_{t_1\times t_2\in T_{I\times J}} |t_2|\right) \\
&= c\left(\sum_{t_1\in T_I}\sum_{\{t_2\in T_I : t_1\times t_2\in T_{I\times J}\}} |t_1| + \sum_{t_2\in T_J}\sum_{\{t_1\in T_I : t_1\times t_2\in T_{I\times J}\}} |t_2|\right) \\
&\leq c\left(\sum_{t_1\in T_I} c_{\mathrm{sp}}^r|t_1| + \sum_{t_2\in T_J} c_{\mathrm{sp}}^c|t_2|\right) \leq c\; c_{\mathrm{sp}}\cdot\left(L(T_I)|I| + L(T_J)|J|\right).
\end{aligned}$$

Damit ist die Abschätzung (4.16) bewiesen. □

4.2.3.2 Geometrische Konstruktion von $T_{I\times I}^{\mathrm{geo}}$

Zur Konstruktion von $T_{I\times I}^{\mathrm{geo}}$ muss nach (4.14) die Zulässigkeitsbedingung (4.13) ausgewertet werden. Dazu müssen Durchmesser und Abstände berechnet werden. Zur Berechnung des Durchmessers eines Clusters t sind $\mathcal{O}\left(|t|^2\right)$ Abstandsbestimmungen notwendig. Der Abstand zwischen zwei Clustern t_1 und t_2 kann geometrisch mit einem Aufwand von $\mathcal{O}\left(|t_1|\cdot|t_2|\right)$ berechnet werden. Um den Aufwand zu reduzieren werden statt der exakten Werte für die Durchmesser von t_1 bzw t_2 und den Abstand zwischen t_1 und t_2 Näherungen berechnet.

Erfolgt die Clusterung in regelmäßige Teilquader, dann lassen sich die Durchmesser von t_1 bzw. t_2 durch die Durchmesser der entsprechenden Teilquader approximieren. Analog wird der Abstand zwischen t_1 und t_2 durch den Abstand der Teilquader um t_1 bzw. t_2 approximiert.

Im Fall der Clusterung mit Hilfe der Hauptkomponenten wird der Abstand der Cluster unter Berücksichtigung der Radien der Cluster approximiert.

4.3 Hierarchische Matrizen: Operationen und Analyse

In diesem Abschnitt wollen wir neben der Definition der hierarchischen Matrizen die arithmetischen Operationen Matrix-Vektor-Multiplikation (MVM), Matrix-Addition sowie die Matrix-Multiplikation von $\mathcal{H}$-Matrizen skizzieren und deren asymptotische Komplexität ermitteln. Die beiden letztgenannten Operationen werden zur Berechnung der $\mathcal{H}$-Inversen und zur Berechnung der $\mathcal{H}$-LU-Zerlegung verwendet.

4.3.1 Die Menge der hierarchischen Matrizen

Definition 4.16. Die Menge der hierarchischen Matrizen auf dem Block-Clusterbaum $T_{I\times J}$ mit der zulässigen Partition $P = \mathcal{L}(T_{I\times J})$ und blockweisem Maximalrang r werden definiert als

$$\mathcal{H}(T_{I\times J}, r) := \{A\in\mathbb{C}^{I\times J} : \operatorname{rank} A_b \leq r, \quad \text{für alle zulässigen} \quad b = t\times s\in P\}.$$

Lemma 4.17. *Sei $A \in \mathcal{H}(T_{I\times J}, r)$. Dann gilt für den Speicherbedarf*

$$\mathcal{N}_{\text{Speicher}}(A) \leq c_{\text{sp}} \max\{r, n_{\min}\} \left(L(T_I)|I| + L(T_J)|J|\right).$$

Beweis. Da die Voraussetzungen von Lemma 4.15 erfüllt sind, folgt die Aussage aus dem Lemma. □

4.3.2 Matrix-Vektor-Multiplikation

Sei $A \in \mathcal{H}(T_{I\times J}, r)$. Die MVM wird blockweise realisiert, $A\boldsymbol{x} = \sum_{t_1\times t_2 \in \mathcal{L}(T_{I\times J})} A_{t_1\times t_2}\boldsymbol{x}_{t_2}$. Liegt der Block $t_1 \times t_2$ im vollen Format vor, dann wird die elementweise MVM (Alg. 7) eingesetzt. Mit (4.17) benötigt die MVM

$$|t_1|(2|t_2| - 1) \leq 2n_{\min}(|t_1| + |t_2|) - |t_1|$$

Operationen. Liegt der Block $t_1 \times t_2$ im Format (4.1) vor, dann wird Alg. 8 angewendet. Die MVM benötigt dann $2r(|t_1| + |t_2|) - r - |t_1|$ Operationen. Der maximale Aufwand zur MVM eines Blocks mit einem Subvektor ist daher $2 \cdot \max\{r, n_{\min}\}(|t_1| + |t_2|)$. Eine Abschätzung für den Gesamtaufwand ergibt sich analog zum Beweis von Lemma 4.15 durch Summation über alle blockweisen Matrix-Vektor-Multiplikationen:

$$\mathcal{W}_{\text{MVM}} \leq 2\, c_{\text{sp}}\, \max\{r, n_{\min}\} \left(L(T_I)|I| + L(T_J)|J|\right).$$

Im Gegensatz zur Addition und Multiplikation von $\mathcal{H}$-Matrizen und darauf aufbauenden Operationen ist die MVM exakt bis auf Maschinengenauigkeit. Eine Parallelisierung der MVM wird in [13] beschrieben.

Algorithmus 11 $\mathcal{H}$-Matrix-Vektor-Multiplikation ($T_{I\times J}$, A, $\boldsymbol{x}$, $\boldsymbol{y}$)

```
1: if t_1 × t_2 ∈ L(T_{I×J}) then
2:     if A_{t_1×t_2} hat Form (4.1) then
3:         MVM für Format (4.1) (|t_1|, |t_2|, r, U, V^H, x_{t_2}, y_{t_1})
4:     else
5:         MVM (|t_1|, |t_2|, A_{t_1×t_2}, x_{t_2}, y_{t_1})
6: else
7:     for all t'_1 ∈ S(t_1) do
8:         for all t'_2 ∈ S(t_2) do
9:             H-Matrix-Vektor-Multiplikation (T_{t'_1×t'_2}, A, x)
```

4.3.3 Blockweise und globale Normen

Liegt A als $\mathcal{H}$-Matrix vor, so gilt

$$\|A\|_F = \left(\sum_{b\in\mathcal{L}(T_{I\times J})} \|A_b\|_F^2 \right)^{\frac{1}{2}}.$$

Lemma 4.18. *Zur Berechnung der Frobeniusnorm von $A \in \mathcal{H}(T_{I\times J}, r)$ werden höchstens*

$$\mathcal{W}_{\|\cdot\|_F} \leq c_{\mathrm{sp}} \min\{n_{\min}, 2r^2\} \left(L(T_I)|I| + L(T_J)|J|\right)$$

Operationen benötigt.

Beweis. Der Beweis kann analog zum Beweis des Lemmas 4.15 geführt werden. Lediglich die Konstante muss ersetzt werden. □

Die Spektralnorm einer $\mathcal{H}$-Matrix ist dagegen vergleichsweise schwierig zu berechnen. In [33] wird die näherungsweise Berechnung durch ein iteratives Verfahren vorgeschlagen. Dabei ist die Wahl des Startvektors problematisch. Zudem ist die Angabe eines geeigneten Abbruchkriteriums schwierig. Wir beschränken uns auf die Angabe von Abschätzungen.

Lemma 4.19. *Gegeben sei die $m \times n$-Blockmatrix*

$$A = \begin{bmatrix} A_{11} & \dots & A_{1\ell} \\ \vdots & & \vdots \\ A_{\ell 1} & \dots & A_{\ell\ell} \end{bmatrix},$$

wobei $A_{ij} \in \mathbb{C}^{m_i \times n_j}$ und $\sum_{i=1}^{\ell} m_i = m$ sowie $\sum_{j=1}^{\ell} n_j = n$. Dann gilt

$$\max_{1\leq i,j\leq \ell} \|A_{ij}\|_2 \leq \|A\|_2 \leq \left(\max_{1\leq i\leq \ell} \sum_{j=1}^{\ell} \|A_{ij}\|_2^2\right)^{\frac{1}{2}} \cdot \left(\max_{1\leq j\leq \ell} \sum_{i=1}^{\ell} \|A_{ij}\|_2^2\right)^{\frac{1}{2}}. \tag{4.18}$$

Satz 4.20. *Sei $P = \mathcal{L}(T_{I\times J})$ eine zulässige Partitionierung der Matrix A. Dann gilt*

$$\max_{b\in\mathcal{L}(T_{I\times J})} \|A_b\|_2 \leq \|A\|_2 \leq c_{\mathrm{sp}}\, L(T_{I\times J}) \max_{b\in\mathcal{L}(T_{I\times J})} \|A_b\|_2 . \tag{4.19}$$

Beweis. Die Beziehung $\max_{b\in\mathcal{L}(T_{I\times J})} \|A_b\|_2 \leq \|A\|_2$ folgt aus dem ersten Teil der Abschätzung (4.18). Zum Beweis der Abschätzung $\|A\|_2 \leq c_{\mathrm{sp}} L(T_{I\times J}) \max_{b\in\mathcal{L}(T_{I\times J})} \|A_b\|_2$ sei A_ℓ der Teil der Matrix, dessen Blöcke Blätter der ℓ-ten Ebene des Block-Clusterbaums sind, d.h.

$$(A_\ell)_b = \begin{cases} A_b & \text{falls } b \in \mathcal{L}(T_{I\times J}^{(\ell)}), \\ 0 & \text{sonst.} \end{cases}$$

Da A_ℓ pro Zeile bzw. Spalte höchstens c_{sp} Blöcke besitzt, gilt $\|A_\ell\|_2 \leq c_{\mathrm{sp}} \max_{b\in\mathcal{L}(T_{I\times J}^{(\ell)})} \|A_b\|_2$.

$$\|A\|_2 \leq \sum_{\ell=1}^{L(T_{I\times J})} \|A_\ell\|_2 \leq \sum_{\ell=1}^{L(T_{I\times J})} c_{\mathrm{sp}} \max_{b\in\mathcal{L}(T_{I\times J}^{(\ell)})} \|A_b\|_2 \overset{\text{Teil 2 von (4.18)}}{\leq} c_{\mathrm{sp}} L(T_{I\times J}) \max_{b\in\mathcal{L}(T_{I\times J})} \|A_b\|_2 .$$

□

4.3.4 Addition

Die Addition zweier hierarchischer Matrizen $A_{\mathcal{H}} \in \mathcal{H}(T_{I\times J}, r_A)$ und $B_{\mathcal{H}} \in \mathcal{H}(T_{I\times J}, r_B)$ ist blockweise definiert. Vollbesetzte Blöcke werden elementweise addiert. Da die Addition von Niedrigrang-Blöcken nicht abgeschlossen ist, liegt das exakte Ergebnis in der Menge $\mathcal{H}(T_{I\times J}, r_A + r_B)$. Benutzt man zur Addition der Niedrigrang-Blöcke die in Abschnitt 4.1.6 definierte approximative Addition, die im Folgenden mit $\oplus_{r_{\max}}$ bezeichnet wird, und setzt $r = r_A = r_B$, so liegt das Ergebnis der Addition in $\mathcal{H}(T_{I\times J}, r)$. Die Analyse von Alg. 12 zeigt, dass der Aufwand $\mathcal{O}\left(r^2\left(|I|\log|I| + |J|\log|J|\right)\right)$ ist.

Algorithmus 12 $\mathcal{H}$-Matrix-Add ($T_{I\times J}$, A, B, C, $r_{\max}$)

```
1: if t_1 × t_2 ∈ L(T_{I×J}) then
2:     if A_{t_1×t_2} und B_{t_1×t_2} haben Form (4.1) then
3:         C_{t_1×t_2} = A_{t_1×t_2} ⊕_{r_max} B_{t_1×t_2}
4:     else
5:         C_{t_1×t_2} = A_{t_1×t_2} + B_{t_1×t_2}
6: else
7:     for all t'_1 ∈ S(t_1) do
8:         for all t'_2 ∈ S(t_2) do
9:             H-Matrix-Add (T_{t'_1×t'_2}, A, B, C, r_max)
```

Um den Gesamtaufwand abzuschätzen, betrachten wir die auftretenden Blocktypen. Sei $b = t_1 \times t_2$ ein Block im Block-Clusterbaum $T_{I\times J}$. Für die Addition zweier Voll-Format-Blöcke werden nach (4.17) $n_{\min}(|t_1| + |t_2|)$ Operationen benötigt. Mit der Abschätzung $r_{A|_b} + r_{B|_b} \leq 2r_{\max}$ erhält man für die Kosten der Addition zweier Niedrigrang-Blöcke nach Satz 4.3 den Ausdruck $173\frac{1}{3}r_{\max}^3 + 24r_{\max}^2(|t_1| + |t_2|)$.

Der Gesamtaufwand einer Matrix-Matrix-Addition im hierarchischen Format ist daher

$$\begin{aligned}
\mathcal{W}_{A_{\mathcal{H}} + B_{\mathcal{H}}} &= \sum_{t_1\times t_2 \in T_{I\times J}} \mathcal{W}_{A_{|t_1\times t_2} + B_{|t_1\times t_2}} \\
&\leq \sum_{t_1\times t_2 \in T_{I\times J}} \left[173\frac{1}{3}r_{\max}^3 + \max\{24r_{\max}^2, n_{\min}\}\left(|t_1| + |t_2|\right)\right] \\
&= 173\frac{1}{3}r_{\max}^3 c_{\mathrm{sp}} \min\{|T_I|, |T_J|\} + \max\{24r_{\max}^2, n_{\min}\} c_{\mathrm{sp}}\left(L(T_I)|I| + L(T_J)|J|\right).
\end{aligned}$$

Der durch die approximative Addition entstehende Fehler kann zum Verlust wichtiger Matrixeigenschaften wie der positiven Definitheit führen. Kleine Eigenwerte können durch die Rundung verschwinden und die Matrix wird indefinit. Mit der in [12] vorgestellten Methode kann die positive Definitheit des Resultats der $\mathcal{H}$-Matrix-Addition garantiert werden.

4.3.5 Multiplikation von $\mathcal{H}$-Matrizen

In der Definition der Matrix-Matrix-Multiplikation $A \cdot B$ für zwei vollbesetzte Matrizen A, B wird vorausgesetzt, dass die Anzahl der Spalten von A mit der Anzahl der Zeilen

von B identisch ist. Daher wird für die Definition der hierarchischen Matrix-Matrix-Multiplikation vorausgesetzt, dass $A \in \mathcal{H}(T_{I\times J}, r_A)$ und $B \in \mathcal{H}(T_{J\times K}, r_B)$.

Das exakte Produkt zweier $\mathcal{H}$-Matrizen ist aber im Allgemeinen nicht in der Blockpartition $\mathcal{L}(T_{I\times K})$ darstellbar. Im folgenden Abschnitt wird der Produkt-Block-Clusterbaum definiert, welcher das exakte Produkt aufnehmen kann.

4.3.5.1 Produkt-Block-Clusterbaum

Definition 4.21. Der Produkt-Block-Clusterbaum T_{IJK} von $T_{I\times J}$ und $T_{J\times K}$ ist induktiv definiert durch

1. $I \times K$ ist die Wurzel von T_{IJK} und
2. die Menge der Söhne des Blocks $t_1 \times t_2 \in T_{IJK}$ der ℓ-ten Ebene von T_{IJK} ist

$$S_{IJK}(t_1 \times t_2) := \{t_1' \times t_2' : \exists r \in T_J^{(\ell)}, r' \in T_J^{(\ell+1)} : \\ t_1' \times r' \in S_{I\times J}(t_1 \times r) \text{ und } r' \times t_2' \in S_{J\times K}(r \times t_2)\}. \tag{4.20}$$

Im Abschnitt 4.2.3, in Definition 4.14, wird die Konstante $c_{\mathrm{sp}}(T_{IJK})$ definiert, die für Komplexitätsabschätzungen nützlich ist. Die maximale Anzahl von Blöcken pro Zeile oder Spalte kann sich im Produkt erhöhen, d.h. eine feinere Partition kann nötig werden.

Lemma 4.22. *Der Produkt-Block-Clusterbaum ist ein Block-Clusterbaum zu den Clusterbäumen T_I und T_K. Zudem gilt*

$$L(T_{IJK}) \leq \min\{L(T_{I\times J}), L(T_{J\times K})\}$$

und

$$c_{\mathrm{sp}}(T_{IJK}) \leq c_{\mathrm{sp}}(T_{I\times J}) \cdot c_{\mathrm{sp}}(T_{J\times K}).$$

Beweis. Zur vollständigen Darstellung geben wir den Beweis aus [9] wieder.

Die Sohnabbildung (4.20) basiert auf den Sohnabbildungen $S_{I\times J}$ und $S_{J\times K}$. Bilden $S_{I\times J}$ oder $S_{J\times K}$ auf die leere Menge ab, dann bildet auch S_{IJK} auf die leere Menge ab. Daher gilt $L(T_{I\times K}) \leq \min\{L(T_{I\times J}), L(T_{J\times K})\}$.

Wegen der Definition 4.14 gilt

$$\begin{aligned} c_{\mathrm{sp}}^{(r)}(T_{IJK}, t_1) &= |\{t_2 \in T_K : t_1 \times t_2 \in T_{IJK}\}|, \\ c_{\mathrm{sp}}^{(c)}(T_{IJK}, t_2) &= |\{t_1 \in T_I : t_1 \times t_2 \in T_{IJK}\}|, \\ c_{\mathrm{sp}}(T_{IJK}) &= \max\{\max_{t_1\in T_I} c_{\mathrm{sp}}^{(r)}(T_{IJK}, t_1), \max_{t_2\in T_K} c_{\mathrm{sp}}^{(c)}(T_{IJK}, t_2)\}. \end{aligned}$$

Aus der Defintion von T_{IJK} folgt

$$\{t_2 \in T_K : t_1 \times t_2 \in T_{IJK}\} \subset \{t_2 \in T_K : \exists r \in T_J : t_1 \times r \in T_{I\times J} \text{ und } r \times t_2 \in T_{J\times K}\}.$$

Weiterhin gilt

$$\begin{aligned}|\{t_2 \in T_K : t_1 \times t_2 \in T_{IJK}\}| &\leq |\{t_2 \in T_K : \exists r \in T_J : t_1 \times r \in T_{I\times J} \text{ und } r \times t_2 \in T_{J\times K}\}| \\ &= \sum_{r\in T_J, t_1\times r\in T_{I\times J}} |\{t_2 \in T_K : r \times t_2 \in T_{J\times K}\}| \\ &\leq c_{\mathrm{sp}}(T_{I\times J})\ c_{\mathrm{sp}}(T_{J\times K}).\end{aligned}$$

□

Sei $t_1\times t_2$ ein Blatt der ℓ-ten Stufe des Produkt-Block-Clusterbaums. Es gilt $(AB)_{t_1\times t_2} = A_{t_1\times J}\, B_{J\times t_2}$. Im Folgenden wird gezeigt, wie sich die Einträge des Blocks aus der Summe von Produkten von Matrixblöcken ergibt.

Dazu bezeichnen wir den Vorgänger eines Clusters $t_1 \in T_I^{(\ell)}$ in der j-ten Stufe mit $F^{(j)}(t_1)$ und analog den Vorgänger eines Clusters $t_2 \in T_K^{(\ell)}$ in der j-ten Stufe mit $F^{(j)}(t_2)$ und definieren die Menge

$$\begin{aligned}U^{(j)}(t_1 \times t_2) := \{r \in T_J^{(j)} : &F^{(j)}(t_1) \times r \in T_{I\times J} \text{ und } r \times F^{(j)}(t_2) \in \mathcal{L}(T_{J\times K}) \\ &\text{oder } F^{(j)}(t_1) \times r \in \mathcal{L}(T_{I\times J}) \text{ und } r \times F^{(j)}(t_2) \in T_{J\times K}\}.\end{aligned} \tag{4.21}$$

Das folgende Lemma zeigt, dass die gesamte Indexmenge J zur Multiplikation berücksichtigt wird.

Lemma 4.23. *Es gilt*

$$\bigcup_{j=0}^{\ell} \bigcup_{r\in U^{(j)}(t_1\times t_2)} r = J, \tag{4.22}$$

wobei die Vereinigung paarweise disjunkt ist. Zudem gilt

$$|U^{(j)}(t_1 \times t_2)| \leq \min\{c_{\mathrm{sp}}(T_{I\times J}), c_{\mathrm{sp}}(T_{J\times K})\},\ 0 \leq j \leq \ell. \tag{4.23}$$

Beweis. Zur vollständigen Darstellung führen wir den Beweis aus [9] aus.

Sei $r' \in T_J$. Es gilt $b_{I\times J}^{(0)} = F^{(0)}(t_1) \times J \in T_{I\times J}$ und $b_{J\times K}^{(0)} = J \times F^{(0)}(t_2) \in T_{J\times K}$. Sind weder $b_{I\times J}^{(0)}$ noch $b_{J\times K}^{(0)}$ Blätter in T_{IJK}, dann existiert ein Cluster $r^{(1)} \in T_J$, so dass $r' \subset r^{(1)}$ und $b_{I\times J}^{(1)} = F^{(1)}(t_1) \times r^{(1)} \in T_{I\times J}$ und $b_{J\times K}^{(1)} = r^{(1)} \times F^{(1)}(t_2) \in T_{J\times K}$ gilt. Wir steigen solange in den Block-Clusterbäumen $T_{I\times J}$ und $T_{J\times K}$ ab, bis $b_{I\times J}^{(j)} = F^{(j)}(t_1) \times r^{(j)} \in \mathcal{L}(T_{I\times J})$ oder $b_{J\times K}^{(j)} = r^{(j)} \times F^{(j)}(t_2) \in \mathcal{L}(T_{J\times K})$ gilt, wobei $r' \subset r^{(j)}$. Daher ist $r' \subset r^{(j)} \in U^{(j)}$. Da $t_1 \times t_2 \in \mathcal{L}(T_{IJK})$ folgt $j \leq \ell$.

Da $U^{(j)}(t_1 \times t_2)$ aus der j-ten Ebene von T_J konstruiert wird, sind die Elemente von $U^{(j)}(t_1\times t_2)$ paarweise verschieden. Seien $r \in U^{(j)}(t_1\times t_2)$ und $r^* \in U^{(j^*)}(t_1\times t_2)$, $j \leq j^*$, und $r \cap r^* \neq \emptyset$. Da $r, r^* \in T_J$ folgt $r^* \subset r$. Daher ist

$$F^{(j^*)}(t_1) \times r^* \subset F^{(j)}(t_1) \times r \text{ und } r^* \times F^{(j^*)}(t_2) \subset r \times F^{(j)}(t_2). \tag{4.24}$$

Aus der Definition von $U^{(j)}(t_1 \times t_2)$ ergibt sich, dass $F^{(j)}(t_1) \times r^* \in \mathcal{L}(T_{I\times J})$ oder $r^* \times F^{(j)}(t_2) \in \mathcal{L}(T_{J\times K})$ ist. Daher ist eine der Inklusionen (4.24) eine Gleichheit. Daraus folgt $r = r^*$. Aus

$$|U^{(j)}(t_1 \times t_2)| \le |\{r \in T_J^{(j)} : F^{(j)}(t_1) \times r \in T_{I\times J}\}| \le c_{\mathrm{sp}}(T_{I\times J})$$

und

$$|U^{(j)}(t_1 \times t_2)| \le |\{r \in T_J^{(j)} : r \times F^{(j)}(t_2) \in T_{J\times K}\}| \le c_{\mathrm{sp}}(T_{J\times K})$$

folgt

$$|U^{(j)}(t_1 \times t_2)| \le \min\{c_{\mathrm{sp}}(T_{I\times J}), c_{\mathrm{sp}}(T_{J\times K})\}.$$

□

Für das Produkt AB, eingeschränkt auf den Block $t_1 \times t_2$, gilt

$$(AB)_{t_1\times t_2} := \sum_{j=0}^{\ell} \sum_{r\in U_j(t_1\times t_2)} A_{t_1\times r} B_{r\times t_2}. \tag{4.25}$$

Satz 4.24. *Seien $A \in \mathcal{H}(T_{I\times J}, r_A)$ und $B \in \mathcal{H}(T_{J\times K}, r_B)$. Dann ist das Produkt $AB \in \mathcal{H}(T_{IJK}, r)$, wobei*

$$r \le L(T_{IJK}) \min\{c_{\mathrm{sp}}(T_{I\times J}), c_{\mathrm{sp}}(T_{J\times K})\} \max\{r_A, r_B, n_{\min}\}.$$

Zur Berechnung des Produkts benötigt man höchstens

$$c_{\mathrm{sp}}(T_{IJK}) L(T_{IJK}) \max\{r'_A \mathcal{W}_{\mathrm{MVM}}(A), r'_B \mathcal{W}_{\mathrm{MVM}}(B)\}$$

Operationen, wobei $r'_A = \max\{r_A, n_{\min}\}$ und $r'_B = \max\{r_B, n_{\min}\}$ und $\mathcal{W}_{\mathrm{MVM}}$ der Aufwand der Matrix-Vektor-Multiplikation ist.

Beweis. Der Beweis stammt aus [9] und wird der vollständigen Darstellung wegen wieder gegeben.

Sei $t_1 \times t_2 \in \mathcal{L}(T_{IJK})$ ein Block in der ℓ-ten Stufe von T_{IJK}. Mit der Gleichung (4.25) wird $(AB)_{t_1\times t_2}$ als Summe über $L(T_{IJK}) \cdot \max_{j=0,\dots,\ell} |U^{(j)}(t_1 \times t_2)|$ Matrixprodukte ausgedrückt. Aus (4.21) folgt, dass einer der Faktoren ein Blatt ist. Der Rang des Produkts ist daher höchstens $\max\{r_A, r_B, n_{\min}\}$. Daher folgt

$$r \le L(T_{IJK}) \max_{j=0,\dots,\ell} \left|U^{(j)}(t_1 \times t_2)\right| \max\{r_A, r_B, n_{\min}\},$$

und mit Lemma 4.23 gilt:

$$r \le L(T_{IJK}) \min\{c_{\mathrm{sp}}(T_{I\times J}), c_{\mathrm{sp}}(T_{J\times K})\} \max\{r_A, r_B, n_{\min}\}.$$

Das Produkt $A_{t_1\times r}B_{r\times t_2}$ kann durch Matrix-Vektor-Multiplikationen berechnet werden. Die Kosten dafür sind $\max\{r'_A\mathcal{W}_{\mathrm{MVM}}(A_{t_1\times r}), r'_B\mathcal{W}_{\mathrm{MVM}}(B_{r\times t_2})\}$. Durch Summation ergibt sich:

$$\begin{aligned}
\mathcal{W}_{\mathrm{MMM}}(A,B) &\leq \sum_{t_1\times t_2\in\mathcal{L}(T_{IJK})} \sum_{j=0}^{L(T_{IJK})} \sum_{r\in U^{(j)}(t_1\times t_2)} \max\{r'_A\mathcal{W}_{\mathrm{MVM}}(A_{t_1\times r}), r'_B\mathcal{W}_{\mathrm{MVM}}(B_{r\times t_2})\} \\
&\leq \sum_{t_1\times t_2\in\mathcal{L}(T_{IJK})} \max\{r'_A\mathcal{W}_{\mathrm{MVM}}(A_{t_1\times J}), r'_B\mathcal{W}_{\mathrm{MVM}}(B_{J\times t_2})\} \\
&\leq \sum_{i=0}^{L(T_{IJK})} \sum_{t_1\times t_2\in\mathcal{L}(T_{IJK})\cap T_{IJK}^{(i)}} \max\{r'_A\mathcal{W}_{\mathrm{MVM}}(A_{t_1\times J}), r'_B\mathcal{W}_{\mathrm{MVM}}(B_{J\times t_2})\} \\
&\leq c_{\mathrm{sp}}(T_{IJK})\ L(T_{IJK})\ \max\{r'_A\mathcal{W}_{\mathrm{MVM}}(A_{t_1\times J}), r'_B\mathcal{W}_{\mathrm{MVM}}(B_{J\times t_2})\}.
\end{aligned}$$

□

4.3.5.2 Erhalten der Blockstruktur

Definition 4.25. Sei $T_{I\times I}$ der aus dem Clusterbaum T_I konstruierte Block-Clusterbaum. Die *Idempotenzkonstante* c_{id} ist definiert als

$$c_{\mathrm{id}}(b) := |\{t'_1\times t'_2\in T_{I\times I} : t'_1\times t'_2\subset b \text{ und } \exists r'\in T_I \text{ mit } t'_1\times r', r'\times t'_2\in T_{I\times I}\}|$$

für $b\in\mathcal{L}(T_{I\times I})$ und

$$c_{\mathrm{id}} := \max_{b\in\mathcal{L}(T_{I\times I})} c_{\mathrm{id}}(b). \tag{4.26}$$

Falls T_{III} nicht feiner als $T_{I\times I}$ ist, so ist $c_{\mathrm{id}} = 1$.

Satz 4.26. *Seien $T_{I\times I}$ ein Block-Clusterbaum der Tiefe $L(T_{I\times I})$ und $A, B\in\mathcal{H}(T_{I\times I}, r)$. Dann ist das Produkt $AB\in\mathcal{H}(T_{I\times I}, \hat{r})$ mit*

$$\hat{r}\leq c_{\mathrm{id}}\ c_{\mathrm{sp}}\ L(T_{I\times I})\ \max\{r, n_{\min}\}.$$

Beweis. Nach Satz 4.24 ist $AB\in\mathcal{H}(T_{III}, r^*)$ mit $r^*\leq L(T_{III})\, c_{\mathrm{sp}}(T_{I\times I})\, \max\{r, n_{\min}\}$. Ist ein Blatt von $T_{I\times I}$ in einem Blatt von T_{III} enthalten, so wird der Rang durch die Beschränkung auf das Blatt von $T_{I\times I}$ nicht erhöht. Enthält ein Blatt von $T_{I\times I}$ Blätter von T_{III}, dann ist deren Anzahl durch c_{id} beschränkt. Daher ist in diesem Fall der Rang durch $r^*\, c_{\mathrm{id}}$ beschränkt. □

Daher sollte c_{id} unabhängig von der Problemgröße beschränkt sein. Für reguläre Finite-Elemente-Gitter und daraus konstruierte Partitionen ist $c_{\mathrm{id}}\leq (c_g c_G c_\Omega)^2(2+\eta)^d$ beschränkt, wobei die Konstanten c_g, c_G und c_Ω durch die Geometrie bzw. das Gitter bestimmt sind und d die räumliche Dimension bezeichnet. Die Konstante η stammt aus der Zulässigkeitsbedingung (4.13). Einen Beweis enthält [9].

4.3.5.3 Gerundete Multiplikation

Da die Komplexität der Matrixoperationen vom Rang abhängt, der Rang aber bei der MMM anwächst, nimmt die Zeit zur Ausführung nachfolgender Operationen zu. Falls das Produkt $AB \in \mathcal{H}(T_{I\times I}, r)$ durch eine Matrix aus $\mathcal{H}(T_{I\times I}, r')$, $r' < r$, approximiert werden soll, kann der Rang durch den Einsatz der in Abschnitt 4.1.6 eingeführten gerundeten Addition in (4.25) reduziert werden.

Der folgende Teile-und-Herrsche Ansatz ist schneller als obige Methode. Die Idee beruht auf der blockweisen MMM. Zu Begin des Verfahrens ist $C \in \mathcal{H}(T_{I\times K}, r')$ die Nullmatrix. Die Aktualisierung geschieht blockweise durch $C = C + AB$. Seien $A \in \mathcal{H}(T_{I\times J}, r_A)$, $B \in \mathcal{H}(T_{J\times K}, r_B)$. Diese besitzen, entsprechend ihren Block-Clusterbäumen, die Blockstruktur

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad \text{und} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}.$$

Das Produkt AB hat dann die Blockstruktur

$$\begin{bmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{21} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{21} + A_{22}B_{22} \end{bmatrix}.$$

Die einzelnen Produkte $A_{ik}B_{kj}$, $i, j, k = 1, 2$, bestehen dabei nur aus halb sovielen Zeilen bzw. Spalten als die Ausgangsmatrix und können daher mit geringerem Aufwand berechnet werden. Besitzt die Zielmatrix C Söhne in $T_{I\times K}$, dann ergibt sich

$$C_{ij} = C_{ij} + A_{i1}B_{1j} + A_{i2}B_{2j}, \quad i, j = 1, 2.$$

Ist C_{ij} ein Blatt in $T_{I\times K}$, dann wird die Summe $A_{i1}B_{1j} + A_{i2}B_{2j}$ zu einer Rang-r'-Matrix R_{ij}, $i, j = 1, 2$, gerundet. Der Block

$$\begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix}$$

wird durch Agglomeration (siehe Abschnitt 4.1.7) zu einer Rang-r'-Matrix zusammen gefasst, bevor er durch gerundete Addition zu C_{ij} addiert wird. Die Matrix-Multiplikation wird durch Alg. 13 lediglich schematisch dargestellt. Zur vollständigen Spezifikation werden die Implementierungen für die Funktionen $\mathrm{MMM}^{\mathcal{H}\times\mathcal{H}\to M}$, $\mathrm{MMM}^{\mathcal{H}\times M\to\mathcal{H}}$ sowie $\mathrm{MMM}^{M\times\mathcal{H}\to\mathcal{H}}$ benötigt. Die verschiedenen MMM sind selbst wieder rekursive Funktionen. Die Matrix-Matrix-Multiplikationen $\mathrm{MMM}^{\mathcal{H}\times M\to\mathcal{H}}$ sowie $\mathrm{MMM}^{M\times\mathcal{H}\to\mathcal{H}}$ enthalten zudem Fallunterscheidungen bzgl. des Formats (volles Format oder Format (4.1)) des übergebenen Matrixblocks. Die technischen Details können in der Implementierung[2] eingesehen werden.

Eine alternative algorithmische Umsetzung der MMM findet man in [39]. Die Komplexitätsanalyse in [40] zeigt, dass die MMM mit $\mathcal{O}\left(r^2\, L^2(T_I)\,|I| + r^3|I|\right)$ Operationen durchgeführt werden kann. Eine parallele Variante der hierarchischen MMM wird in [50] eingeführt.

[2] http://bebendorf.ins.uni-bonn.de/AHMED.html

Algorithmus 13 $\mathcal{H}$-MMM $(t_1 \times t_2,\, C,\, t_1 \times q,\, A,\, q \times t_2,\, B)$

```
1:  if t_1 × t_2 ∈ L(T_{I×K}) then
2:      MMM^{H×H→M} (t_1 × t_2, C, t_1 × q, A, q × t_2, B)
3:  else
4:      if t_1 × q ∉ L(T_{I×J}) and q × t_2 ∉ L(T_{J×K}) then
5:          for all t'_1 ∈ S_I(t_1) do
6:              for all t'_2 ∈ S_K(t_2) do
7:                  for all q' ∈ S_J(q) do
8:                      H-MMM (t'_1 × t'_2, C, t'_1 × q', A, q' × t'_2, B)
9:      else
10:         if t_1 × q ∈ L(T_{I×J}) then
11:             MMM^{M×H→H} (t_1 × t_2, C, t_1 × q, A, q × t_2, B)
12:         else
13:             MMM^{H×M→H} (t_1 × t_2, C, t_1 × q, A, q × t_2, B)
```

4.3.6 Die $\mathcal{H}$-Inverse

Lässt sich eine Matrix A oder eine Approximation von A als $\mathcal{H}$-Matrix darstellen, so muss dies für die Inverse A^{-1} nicht zutreffen. Die Existenz der approximativen Inversen der FE-Matrix im hierarchischen Format wird in [11] bewiesen, für weitere Verallgemeinerungen des Beweises siehe [6, 7].

4.3.6.1 Iteratives Verfahren

Sei $\hat{C}$ die Inverse zu A. Dann gilt $A - \hat{C}^{-1} = 0$. Daraus lässt sich eine Funktion $F : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$, $F(C) := A - C^{-1}$ konstruieren, die bei $\hat{C} = A^{-1}$ eine Nullstelle hat. Diese kann mit dem Newton-Verfahren[3]

$$D\left(F\left(C_{(k)}\right)\right)\left(C_{(k+1)} - C_{(k)}\right) = -F(C_{(k)})$$

bestimmt werden. Dazu wird die Ableitung des oben definierten F an der Stelle C in Richtung V berechnet, $D(F(C))(V) = C^{-1}VC^{-1}$. Wegen $C_{(k)}^{-1}\left(C_{(k+1)} - C_{(k)}\right)C_{(k)}^{-1} = C_{(k)}^{-1} - A$ ergibt sich das Iterationsverfahren

$$C_{(k+1)} = C_{(k)}\left(2I - AC_{(k)}\right).$$

Pro Iteration werden zwei Matrix-Matrix-Multiplikationen benötigt. Die Anzahl der Iterationen hängt von der Wahl des Startwerts $C_{(0)}$ ab.

Unter der Annahme, dass die Inverse im hierarchischen Format darstellbar und der Startwert gut gewählt ist, konvergiert das Newton-Verfahren (mit $\mathcal{H}$-Arithmetik) quadratisch, siehe Lemma 4.20 bzw. Lemma 4.22 in [33].

[3] in diesem Fall auch als Schulze-Verfahren bezeichnet

4.3.6.2 Rekursives Verfahren

Im Allgemeinen ist das folgende rekursive Verfahren effizienter als das iterative Verfahren.

Die Inverse der Matrix

$$A = \begin{bmatrix} A_{t_1t_1} & A_{t_1t_2} \\ A_{t_2t_1} & A_{t_2t_2} \end{bmatrix}$$

ist

$$C := A^{-1} = \begin{bmatrix} A_{t_1t_1}^{-1} + A_{t_1t_1}^{-1} A_{t_1t_2} S^{-1} A_{t_2t_1} A_{t_1t_1}^{-1} & -A_{t_1t_1}^{-1} A_{t_1t_2} S^{-1} \\ -S^{-1} A_{t_2t_1} A_{t_1t_1}^{-1} & S^{-1} \end{bmatrix}$$

wobei $S := A_{t_2t_2} - A_{t_2t_1} A_{t_1t_1}^{-1} A_{t_1t_2}$ das Schur-Komplement von $A_{t_1t_1}$ bzgl. $A_{t_2t_1}$ in A ist. Zur Berechnung von A^{-1} werden die Inversen von $A_{t_1t_1}$ und S benötigt. Liegen $A_{t_1t_1}$ oder S wieder in entsprechender Blockstruktur vor, können die Inversen rekursiv berechnet werden.

Algorithmus 14 BlockInvertiere (t, A, C)

1: **if** $t \in \mathcal{L}(T_I)$ **then**
2: $\quad C_{tt} = A_{tt}^{-1}$
3: **else**
4: $\quad$ BlockInvertiere (t_1, A, C)
5: $\quad T_{t_1t_2} := T_{t_1t_2} - C_{t_1t_1} A_{t_1t_2}$
6: $\quad T_{t_2t_1} := T_{t_2t_1} - A_{t_2t_1} C_{t_1t_1}$
7: $\quad A_{t_2t_2} := A_{t_2t_2} + A_{t_2t_1} T_{t_1t_2}$
8: $\quad$ BlockInvertiere (t_2, A, C)
9: $\quad C_{t_1t_2} := C_{t_1t_2} + T_{t_1t_2} C_{t_2t_2}$
10: $\quad C_{t_2t_1} := C_{t_2t_1} + C_{t_2t_2} T_{t_2t_1}$
11: $\quad C_{t_1t_1} := C_{t_1t_1} + T_{t_1t_2} C_{t_2t_1}$

Die im Alg. 14 aufwändigste Operation ist die Matrix-Matrix-Multiplikation. Die Gesamtlaufzeit wird daher durch diese bestimmt. Werden im Algorithmus die Operationen durch ihre hierarchischen Ausführungen ersetzt, so wird die approximierte Inverse in $\mathcal{H}$-Format berechnet. Das folgende Lemma gilt unter der Annahme, dass das Ergebnis einer MMM im Algorithmus auf den Rang r gerundet werden kann, ohne das der Rundungsfehler stark anwächst.

Lemma 4.27. *Die Berechnung der hierarchischen Inversen $C \in \mathcal{H}(T_{I\times I}, r)$ von $A \in \mathcal{H}(T_{I\times I}, r)$ mit Alg. 14 benötigt $\mathcal{O}\left(r^2 |I| \log^2 |I|\right)$ Operationen.*

Beweis. siehe [33, Satz 5.29] □

4.3.6.3 Aktualisierung der Inversen

Die Lösung einer nichtlinearen partiellen Differentialgleichung mit dem Newton-Verfahren führt auf eine Folge von großdimensionalen linearen Gleichungssystemen. Wegen ihrer Größe werden diese meist iterativ gelöst. Jedoch konvergieren die iterativen Löser bei

schlechter Kondition des Gleichungssystems nur langsam. Deshalb muss in der Regel in jeder Iteration des Newton-Verfahrens ein Vorkonditionierer zur Konvergenzbeschleunigung konstruiert werden. Ändert sich die Matrix des zu lösenden linearen Gleichungssystems nur wenig, so ist eine Aktualisierung der Inversen aus dem vorhergehenden Schritt mit Hilfe der Shermann-Woodbury-Formel möglich. Sei $\tilde{A} = A + \boldsymbol{a}\boldsymbol{b}^T$ und A^{-1} bereits berechnet. Aus dem Ansatz für die Inverse

$$\tilde{A}^{-1} = A^{-1} + \alpha A^{-1}\boldsymbol{a}\boldsymbol{b}^T A^{-1},$$

$\alpha \in \mathbb{R}$, ergibt sich wegen

$$\begin{aligned} I = \tilde{A}\tilde{A}^{-1} &= \left(A + \boldsymbol{a}\boldsymbol{b}^T\right)\left(A^{-1} + \alpha A^{-1}\boldsymbol{a}\boldsymbol{b}^T A^{-1}\right) \\ &= I + \alpha\boldsymbol{a}\boldsymbol{b}^T A^{-1} I + \boldsymbol{a}\boldsymbol{b}^T A^{-1} I + \boldsymbol{a}\boldsymbol{b}^T \alpha A^{-1}\boldsymbol{a}\boldsymbol{b}^T A^{-1} I \\ 0 &= \left(I + \alpha\left(I + \boldsymbol{a}\boldsymbol{b}^T A^{-1}\right)\right)\boldsymbol{a}\boldsymbol{b}^T A^{-1} \\ \alpha &= -\frac{1}{1 + \boldsymbol{b}^T A^{-1}\boldsymbol{a}} \end{aligned}$$

die aktualisierte Inverse

$$\tilde{A}^{-1} = A^{-1} - \frac{A^{-1}\boldsymbol{a}\boldsymbol{b}^T A^{-1}}{1 + \boldsymbol{b}^T A^{-1}\boldsymbol{a}},$$

wobei $\boldsymbol{b}^T A^{-1}\boldsymbol{a} \neq -1$ vorausgesetzt wird.

Die Inverse $\tilde{A}^{-1}$ ergibt sich also aus einer Rang-1-Modifikation von A^{-1}. Liegt A^{-1} im hierarchischen Format vor, so kann die Rang-1-Modifikation durch die im Abschnitt 4.3.4 vorgestellte approximierte Addition erfolgen.

4.3.7 Die $\mathcal{H}$-LU-Zerlegung

Vorausgesetzt, dass der blockweise Rang bei den zur Invertierung benötigten Matrix-Matrix-Multiplikationen nur gering anwächst, hat die Inversion nahezu lineare Komplexität. Es zeigt sich [9], dass die $\mathcal{H}$-LU-Zerlegung in der gleichen asymptotischen Komplexität berechnet werden kann wie die $\mathcal{H}$-Inverse. In numerischen Experimenten hat sich jedoch heraus gestellt, dass die $\mathcal{H}$-LU-Zerlegung mit geringerem Aufwand als die $\mathcal{H}$-Inverse berechnet werden kann. Daher wird zur Vorkonditionierung meist die $\mathcal{H}$-LU-Zerlegung anstatt der $\mathcal{H}$-Inversen angewendet. Zur Simulation der $\mathcal{H}$-Inversen werden hierarchische Versionen der Vorwärts- und Rückwärtssubstitution ((3.8a) bzw. (3.8b)) genutzt.

Aus der Blockstruktur

$$\begin{bmatrix} A_{t_1t_1} & A_{t_1t_2} & \dots & A_{t_1t_N} \\ A_{t_2t_1} & A_{t_2t_2} & \dots & A_{t_2t_N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{t_Nt_1} & A_{t_Nt_2} & \dots & A_{t_Nt_N} \end{bmatrix} = \begin{bmatrix} L_{t_1t_1} & 0 & \dots & 0 \\ L_{t_2t_1} & L_{t_2t_2} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ L_{t_Nt_1} & L_{t_Nt_2} & \dots & L_{t_Nt_N} \end{bmatrix} \times \begin{bmatrix} U_{t_1t_1} & U_{t_1t_2} & \dots & U_{t_1t_N} \\ 0 & U_{t_2t_2} & \dots & U_{t_2t_N} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & U_{t_Nt_N} \end{bmatrix}$$

ergibt sich der Alg. 15 zur Block-LU-Zerlegung durch blockweise Auswertung obiger Gleichung. Werden im Algorithmus die Additionen/Subtraktionen und Multiplikatio-

Algorithmus 15 Block-Gauß-Elimination (t, A, L, U)

1: **if** $t \times t \in \mathcal{L}(T_{I\times I})$ **then**
2: berechne die LU-Zerlegung für das volle Format für den Block $t\times t$: $A_{tt} = L_{tt}U_{tt}$
3: **else**
4: $N = |S_I(t)|$
5: **for** $i = 0 : N-1$ **do**
6: **for** $k = 0 : i$ **do**
7: $A_{t_it_i} := A_{t_it_i} - L_{t_it_k}U_{t_kt_i}$ // Diagonalblöcke von A_{tt} aktualisieren
8: Block-Gauß-Elimination (t_i, A, L, U)
9: **for** $j = i+1 : N-1$ **do**
10: **for** $k = 0 : i$ **do**
11: $A_{t_it_j} := A_{t_it_j} - L_{t_it_k}U_{t_kt_j}$ // Blöcke von $A_{t_it_j}$ aktualisieren ($U_{t_it_j}$)
12: Vorwärtssubstitution: $L_{t_it_i}U_{t_it_j} = A_{t_it_j}$
13: **for** $k = 0 : i$ **do**
14: $A_{t_jt_i} := A_{t_jt_i} - L_{t_jt_k}U_{t_kt_i}$ // Blöcke von $A_{t_jt_i}$ aktualisieren ($L_{t_jt_i}$)
15: Rückwärtssubstitution: $L_{t_jt_i}U_{t_it_i} = A_{t_jt_i}$

nen sowie die Vorwärts- bzw. Rückwärtssubstitution durch entsprechende hierarchische Operationen ersetzt, erhalten wir einen Algorithmus zur $\mathcal{H}$-LU-Zerlegung. Um eine gute Approximation zu garantieren, wird angenommen, dass der blockweise Rang nicht stark ansteigt. Für FE-Matrizen wird die Existenz der Faktoren L und U in [8] nachgewiesen.

Block-Gauß-Eliminations-Algorithmen für binäre Clusterbäume, wie sie üblicherweise durch geometrische Methoden erzeugt werden, sind in [9, 49] zu finden.

Im Fall symmetrisch positiv definiter Matrizen kann die Dreieckszerlegung vereinfacht werden. Die vereinfachte Dreieckszerlegung bezeichnet man als Cholesky-Zerlegung. Durch Ausnutzen der Symmetrie kann bei der Cholesky-Zerlegung etwa die Hälfte des Rechenaufwands gespart werden. Die (hierarchische) Block-Cholesky-Zerlegung kann aus der blockweisen Auswertung von

$$\begin{bmatrix} A_{t_1t_1} & A_{t_1t_2} & \dots & A_{t_1t_N} \\ A_{t_2t_1} & A_{t_2t_2} & \dots & A_{t_2t_N} \\ \vdots & \vdots & \ddots & \vdots \\ A_{t_Nt_1} & A_{t_Nt_2} & \dots & A_{t_Nt_N} \end{bmatrix} = \begin{bmatrix} L_{t_1t_1} & 0 & \dots & 0 \\ L_{t_2t_1} & L_{t_2t_2} & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ L_{t_Nt_1} & L_{t_Nt_2} & \dots & L_{t_Nt_N} \end{bmatrix} \times \begin{bmatrix} L_{t_1t_1} & L_{t_2t_1} & \dots & L_{t_Nt_1} \\ 0 & L_{t_2t_2} & \dots & L_{t_Nt_2} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & L_{t_Nt_N} \end{bmatrix}$$

hergeleitet werden.

5 Algebraische Matrixpartition

Die Laufzeit der arithmetischen Operationen mit $\mathcal{H}$-Matrizen hängt entscheidend von der Matrixpartition ab. Eine geeignete Matrixpartition basiert oft auf einem bzgl. der Kardinalität balancierten Clusterbaum.

Die bisher für die Zerlegung der Indexmenge eingesetzten Partitionierungsalgorithmen liefern für komplexe Geometrien oder adaptive Verfeinerungen einen bzgl. der Anzahl der Freiheitsgrade unbalancierten Clusterbaum und folglich eine stark verbesserbare Matrixpartition.

Deshalb wird in diesem Kapitel eine von der Geometrie des Problems unabhängige Partitionierung der Indexmenge vorgestellt. Im Gegensatz zu Abschnitt 4.2.3 beruht die Konstruktion des Block-Clusterbaums (Abschnitt 5.4) nicht auf Approximationen von geometrischen Abständen bzw. Durchmessern, sondern wertet das in Abschnitt 5.1 eingeführte, rein algebraische, Zulässigkeitskriterium approximativ aus. Teile des Inhalts dieses Kapitels bildeten die Grundlage für den Artikel [10].

5.1 Existenz der $\mathcal{H}$-Inversen auf Grundlage einer algebraischen Matrixpartition

Seien I eine Indexmenge, A eine $I \times I$-Matrix sowie G_A der in (2.5) definierte Matrixgraph. Der Abstand zwischen den Teilmengen $t_1, t_2 \subset I$ ist analog zu (2.2) definiert. Der Durchmesser einer Teilmenge $t \subset I$ ist analog zu (2.4) erklärt. Zudem existieren für eine schwach besetzte Matrix A Konstanten $c_v, d > 0$, so dass die Bedingung

$$|U_\rho(i)| \leq c_v \rho^d \quad \text{für } \rho > 0 \tag{5.1}$$

erfüllt ist, wobei $U_\rho(i) = \{j \in I : d(i,j) \leq \rho\}$ ist.

Sei A eine schwach besetzte, nicht singuläre Matrix. Die Inverse A^{-1} ist im Allgemeinen nicht schwach besetzt. Zur Darstellung der Inversen wollen wir die Neumannsche Reihe verwenden. Zunächst soll der Rang einzelner Blöcke der Inversen abgeschätzt werden. In dem folgenden Lemma wird analysiert, wie sich die Potenzierung auf die Besetzungsstruktur der Matrix auswirkt.

Lemma 5.1. *Sei $\ell \in \mathbb{N}$. Dann gilt*

1. $(A^\ell)_{ij} = 0$ *für $i, j \in I$ mit $d(i,j) > \ell$ und*
2. $[(A^H A)^\ell]_{ij} = 0$ *für $i, j \in I$ mit $d(i,j) > 2\ell$.*

Beweis. Für $\ell = 0, 1$ ist die Aussage offenbar erfüllt. Sei $\ell > 1$ und $\left(A^{\ell-1}\right)_{ik} = 0$ für $d(i,k) > \ell - 1$. Sei nun der Abstand zweier Knoten i, j im Matrixgraphen größer als ℓ.

Da d eine Metrik ist, gilt $\ell < d(i,j) \leq d(i,k) + d(k,j)$. Folglich ist $d(i,k) > \ell - 1$ oder $d(k,j) > 1$ und daher $\left(A^{\ell-1}\right)_{ik} = 0$ oder $(A)_{kj} = 0$. Deswegen gilt

$$\left(A^{\ell}\right)_{ij} = \sum_{k=0}^{n} \left(A^{\ell-1}\right)_{ik} (A)_{kj} = 0. \tag{5.2}$$

Die zweite Aussage ergibt sich analog. □

Seien $t_1, t_2 \subset I$. Wir definieren den *Rand von t_2 bzgl. t_1* als

$$\partial_{t_1} t_2 := \{\nu \in t_2 : \exists i \in t_1 \text{ so dass } d(i,\nu) = \min_{j \in t_2} d(i,j)\},$$

siehe Abb. 5.1.

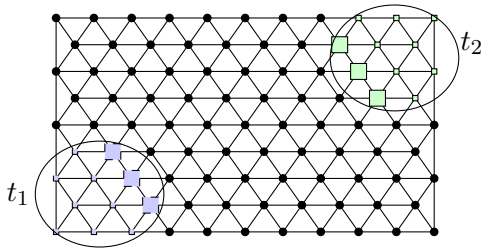


Abb. 5.1: Cluster t_1, t_2 im Matrixgraphen, quadratische Knoten gehören zum Rand

Lemma 5.2. *Seien t_1, t_2 zwei Teilmengen der Indexmenge, die die Bedingung*

$$\operatorname{diam} \partial_{t_1} t_2 \leq \eta \operatorname{dist}(t_1, t_2) \quad \text{für ein} \quad \eta > 0 \tag{5.3}$$

erfüllen. Dann gilt für ein Polynom $p \in \Pi_r$ vom Grad r, dass

1. $\operatorname{rank}[p(A)]_{t_1 t_2} \leq c_v (1+\eta)^d r^d$ *und*
2. $\operatorname{rank}[p(A^H A)]_{t_1 t_2} \leq c_v (1+\eta)^d (2r+1)^d$.

Beweis. Sei $\ell < r$. Wir betrachten ein $i \in L_m := \{i \in t_1 : \operatorname{dist}(i, \partial_{t_1} t_2) = m\}$ für $m > \ell$. Dann ist für $j \in t_2$ nach Lemma 5.1 $(A^{\ell})_{ij} = 0$. Die Mengen L_m sind für $m < \operatorname{dist}(t_1, t_2)$ leer. Folglich besitzen nur die den Elementen der Menge

$$N_\ell = \bigcup_{m=\operatorname{dist}(t_1,t_2)}^{\ell} L_m = \{i \in t_1 : \operatorname{dist}(i, \partial_{t_1} t_2) \leq \ell\}$$

entsprechenden Zeilen Nicht-Null-Einträge. Es gilt $N_\ell \subset N_r$. Da für $r < \operatorname{dist}(t_1, t_2)$ die Menge N_r leer ist, setzen wir $r \geq \operatorname{dist}(t_1, t_2)$ voraus.

$$|N_r| \overset{(5.1)}{\leq} c_v (r + \operatorname{diam}(\partial_{t_1} t_2))^d \overset{(5.3)}{\leq} c_v (r + \eta \operatorname{dist}(t_1, t_2))^d \leq c_v (1+\eta)^d r^d$$

□

Der Rand $\partial_{t_1} t_2$ lässt sich technisch schwierig berechnen. Eine Abschätzung für den Durchmesser eines Clusters kann mit geringerem Aufwand berechnet werden. Deshalb werden wir im Folgenden die im Vergleich zu (5.3) etwas stärkere *algebraische Zulässigkeitsbedingung*

$$\min\{\operatorname{diam} t_1, \operatorname{diam} t_2\} \leq \eta \operatorname{dist}(t_1, t_2), \tag{5.4}$$

verwenden. Die Bedingung (5.3) ist aber aus theoretischer Sicht interessant.

Mit (5.4) ist es möglich, die zur Niedrigrang-Approximation geeigneten Blöcke der Inversen auf rein algebraische Art zu identifizieren. Der Rang eines geeigneten Blocks $[p(A)]_{t_1 t_2}$ kann durch den Grad des Polynoms p abgeschätzt werden.

Unter der Voraussetzung, dass die Matrix gut konditioniert ist, liefert der Satz 5.3 im allgemeinen Fall und der Satz 5.4 für positiv definite Matrizen die Existenz der Inversen im hierarchischen Format. Die numerischen Ergebnisse im Kapitel 6 zeigen jedoch, dass sich die mit der algebraischen Zulässigkeitsbedingung konstruierte Partition zur Berechnung der Faktoren der LU-Zerlegung im hierarchischen Format auch für schlecht konditionierte Matrizen eignet. Sei P_{alg} eine nach Definition 4.5 zulässige Partition, d.h. die Blöcke sind klein oder erfüllen die Zulässigkeitsbedingung (5.4).

Satz 5.3. *Sei A eine nicht singuläre Matrix die* (5.1) *erfüllt. Dann existiert eine Matrix $C_{\mathcal{H}} \in \mathcal{H}(P_{\text{alg}}, r)$ mit $r \sim \left(\frac{|\log \varepsilon|}{|\log \lambda|}\right)^d$, so dass*

$$\left\|A^{-1} - C_{\mathcal{H}}\right\|_2 \leq \operatorname{cond}_2(A)\, \varepsilon \left\|A^{-1}\right\|_2,$$

wobei

$$\lambda := 1 - \frac{1}{\operatorname{cond}_2^2(A)}.$$

Beweis. Da A nicht singulär ist, ist $B := A^H A$ symmetrisch und besitzt nur reelle Eigenwerte. Wegen

$$\langle B\boldsymbol{x}|\boldsymbol{x}\rangle = \left\langle A^H A\boldsymbol{x}|\boldsymbol{x}\right\rangle = \langle A\boldsymbol{x}|A\boldsymbol{x}\rangle = \|A\boldsymbol{x}\|_2^2 \geq 0$$

für $\boldsymbol{x} \in \mathbb{R}^n \backslash \{0\}$ hat B nur positive Eigenwerte. Um die Neumannsche Reihe

$$(I - F)^{-1} = \sum_{j=0}^{\infty} F^j, \quad \|F\|_2 < 1 \tag{5.5}$$

anwenden zu können, skalieren wir mit $\beta := 1/\lambda_{\max}(B)$, und es gilt dann für den Spektralradius $\rho(I - \beta B) = 1 - \frac{\lambda_{\min}(B)}{\lambda_{\max}(B)} < 1$. Setzen wir $F = I - \beta B$, so folgt

$$B^{-1} = \beta \sum_{j=0}^{\infty} (I - \beta B)^j.$$

Mit

$$p_r(B) := \beta \sum_{j=0}^{r-1} (I - \beta B)^j,$$

erhalten wir

$$\left\|B^{-1} - p_r(B)\right\|_2 \leq \beta \sum_{j=r}^{\infty} \rho^j (I - \beta B) \leq \frac{\beta \rho^r (I - \beta B)}{1 - \rho(I - \beta B)} = \left(1 - \frac{\lambda_{\min}(B)}{\lambda_{\max}(B)}\right)^r \left\|B^{-1}\right\|_2$$

und daher

$$\begin{aligned}
\left\|A^{-1} - p_r(B)A^H\right\|_2 &= \left\|(B^{-1} - p_r(B))A^H\right\|_2 \leq \left\|B^{-1} - p_r(B)\right\|_2 \|A\|_2 \\
&\leq \left(1 - \frac{\lambda_{\min}(B)}{\lambda_{\max}(B)}\right)^r \left\|B^{-1}\right\|_2 \|A\|_2 \\
&= \operatorname{cond}_2(A) \left(1 - \frac{1}{\operatorname{cond}_2^2(A)}\right)^r \left\|A^{-1}\right\|_2 .
\end{aligned}$$

Mit Lemma 5.2 folgt für einen Block $b \in P_{\text{alg}}$ der Matrix $C_{\mathcal{H}} := p_r(B)A^H$, dass der Rang höchstens $c_v(1+\eta)^d(2r+1)^d$ ist. □

Für symmetrisch positiv definite Matrizen gestaltet sich die Abschätzung etwas günstiger.

Satz 5.4. *Sei A eine symmetrisch positiv definite Matrix, die* (5.1) *erfüllt. Dann existiert eine Matrix $C_{\mathcal{H}} \in \mathcal{H}(P_{\text{alg}}, r)$ mit $r \sim \left(\frac{|\log \varepsilon|}{|\log \lambda|}\right)^d$ so dass*

$$\left\|A^{-1} - C_{\mathcal{H}}\right\|_2 \leq \varepsilon \left\|A^{-1}\right\|_2 ,$$

wobei

$$\lambda := \frac{\sqrt{\operatorname{cond}_2(A)} - 1}{\sqrt{\operatorname{cond}_2(A)} + 1}.$$

Beweis. Da A symmetrisch positiv definit ist, gilt

$$\left\|A^{-1} - p(A)\right\|_2 = \rho(A^{-1} - p(A)) = \max_{x \in \sigma(A)} |x^{-1} - p(x)|,$$

wobei $\sigma(A)$ das Spektrum von A bezeichnet. Nach Tschebyscheff (siehe [51, Seite 33]) existiert ein Polynom p_r vom Grad r, so dass

$$\left|x^{-1} - p_r(x)\right|_{\infty,[a,b]} \leq c\, \lambda^{r+1}$$

mit $c = \frac{(1+\sqrt{q})^2}{2aq} \leq \frac{2}{a}$, $\lambda = \frac{\sqrt{q}-1}{\sqrt{q}+1}$, $q = \frac{b}{a}$. Seien $a = \left\|A^{-1}\right\|_2^{-1}$ und $b = \|A\|_2$. Da A positiv definit ist, gilt $\sigma(A) \subset [a,b]$ und $a > 0$. Nach Lemma 5.2 hat jeder Block $b \in P_{\text{alg}}$ von $C_{\mathcal{H}} := p_r(A)$ höchstens den Rang $c_v\,(1+\eta)^d\, r^d$. □

Zum Beweis der Sätze wird das Lemma 5.2 genutzt. Im Beweis des Lemmas wird die algebraische Zulässigkeitsbedingung (5.3) verwendet. Folglich ist die stärkere algebraische Zulässigkeitsbedingung (5.4) zum Beweis der Sätze ausreichend. Für die $\mathcal{H}$-Inverse genügt daher ein bzgl. der Kardinalität balancierter Clusterbaum.

Bemerkung 5.5. Das Hauptaugenmerk dieser Arbeit soll auf die Vorkonditionierung iterativer Löser für schwach besetzte Gleichungssysteme mit quadratischer Systemmatrix gerichtet sein. Wir werden uns daher auf die Partitionierung von $I \times I$ statt auf $I \times J$ konzentrieren.

5.2 $\mathcal{H}$-LU-Zerlegung für die algebraische Matrixpartition

In [8] wird die Existenz der Faktoren der $\mathcal{H}$-LU-Zerlegung für FE-Matrizen unter Berücksichtigung einer geometrischen Matrixpartition gezeigt. Im Artikel wird zunächst die Existenz des Schur-Komplements für Blöcke $b \in P_{\text{geo}}$ im hierarchischen Format nachgewiesen. Dazu wird die Existenz der hierarchischen Inversen, siehe [11], ausgenutzt. Sowohl für die Existenz des hierarchischen Schur-Komplements als auch für die Existenz der hierarchischen Inversen von FE-Matrizen werden geometrische Argumente verwendet. Auf die Existenz des hierarchischen Schur-Komplements aufbauend wird mit rein algebraischen Argumenten die Existenz der Faktoren der LU-Zerlegung im hierarchischen Format bewiesen.

Nach Abschnitt 5.1 ist es möglich, den Beweis für die Existenz der $\mathcal{H}$-LU-Zerlegung rein algebraisch durchzuführen.

5.2.1 Schur-Komplement

Die Matrix $A \in \mathbb{R}^{I\times I}$ habe die folgende Blockstruktur

$$A = \begin{bmatrix} A_{tt} & A_{tt'} \\ A_{t't} & A_{t't'} \end{bmatrix}, \tag{5.6}$$

mit $t \subset I$ und $t' := I\backslash t$.

Wir werden sehen, dass jeder Block $b = t_1 \times t_2$ des *Schur-Komplements*

$$S := A_{t't'} - A_{t't}A_{tt}^{-1}A_{tt'},$$

der die algebraische Zulässigkeitsbedingung (5.4) erfüllt, durch einen Matrixblock vom Rang r approximiert werden kann. Dabei hängt der Rang nur logarithmisch sowohl von der Approximationsgenauigkeit ε als auch von $|I|$ ab.

Sei $t \in T_I$. Wir bezeichnen

$$N(t) = \{i \in I : \operatorname{dist}(i,t) \le 1\} \quad \text{und} \quad \mathcal{F}_\eta(t) = \{i \in I : \eta \operatorname{dist}(i,t) \ge \operatorname{diam}(t)\} \tag{5.7}$$

als *Nachbarschaft* bzw. *Fernfeld* von t.

Das folgende Lemma besagt, dass die Nachbarschaft von t_1 im Fernfeld der Nachbarschaft von t_2 ist, falls sich t_1 im Fernfeld von t_2 befindet.

Lemma 5.6. *Seien* $\eta > 0$ *und* $\operatorname{diam}(t_2) \ge 4(1+\eta)$. *Gilt* $t_1 \subset \mathcal{F}_\eta(t_2)$, *dann folgt* $N(t_1) \subset \mathcal{F}_{2\eta}(N(t_2))$.

Beweis. Wegen $\operatorname{diam}(N(t_2)) \le \operatorname{diam}(t_2) + 2$ folgt

$$\begin{aligned}\eta \operatorname{dist}(N(t_1), N(t_2)) &\ge \eta \operatorname{dist}(t_1,t_2) - 2\eta \ge \operatorname{diam}(t_2) - 2\eta \\ &\ge \operatorname{diam}(N(t_2)) - 2 - 2\eta = \left(1 - \frac{2}{\operatorname{diam}(N(t_2))}(1+\eta)\right)\operatorname{diam}(N(t_2)) \\ &\ge \frac{1}{2}\operatorname{diam}(N(t_2)).\end{aligned}$$

□

Mit dem Lemma und unter der Annahme, dass eine Konstante $c > 0$ existiert, so dass

$$\left\|(A_{tt})^{-1}\right\|_2 \leq c \left\|A^{-1}\right\|_2 \tag{5.8}$$

für alle $t \in T_I$ gilt, kann nachgewiesen werden, dass das Schur-Komplement S für jeden Block der Matrix A approximiert werden kann.

Auf Grund der Existenz der Inversen im hierarchischen Format (Sätze 5.3 und 5.4) nehmen wir an, dass für jeden zulässigen Block $t_1 \times t_2$, $t_1, t_2 \subset t$, Matrizen $U \in \mathbb{R}^{t_1 \times r}$, $V \in \mathbb{R}^{t_2 \times r}$, mit $r \sim |\log \varepsilon|^d$, existieren, so dass

$$\left\|(A_{tt}^{-1})_{t_1 t_2} - UV^T\right\|_2 \leq \varepsilon \left\|A_{tt}^{-1}\right\|_2 \tag{5.9}$$

gilt.

Satz 5.7. *Besitze $A \in \mathbb{R}^{I \times I}$ die Blockstruktur* (5.6)*. Dann existiert für das Schur-Komplement*

$$S = A_{t't'} - A_{t't} A_{tt}^{-1} A_{tt'}$$

von A_{tt} in A und jedes $\varepsilon > 0$ eine hierarchische Matrix $S_{\mathcal{H}} \in \mathcal{H}(P_{\text{alg}}, r_S)$, mit Rang $r_S \sim |\log \varepsilon|^d$, so dass

$$\|S - S_{\mathcal{H}}\|_2 < (\log |I|)\, \text{cond}_2(A)\, \varepsilon \, \|A\|_2$$

gilt.

Beweis. Für eine lückenlose Darstellung wird hier der Beweis aus [10] wieder gegeben.

Ziel des Beweises ist es, zu jedem zulässigen (Unter-)Block $t_1 \times t_2 \in P_{\text{alg}}$ des Blocks $A_{t't'}$ und zur vorgegebenen Genauigkeit $\varepsilon > 0$ eine Approximation für den Schur-Komplement-Block $S_{t_1 t_2}$ zu finden.

Falls $\text{dist}(t_1, t_2) \leq 1$ ist, dann ist $\min\{\text{diam}(t), \text{diam}(s)\} \leq \eta$. Daher ist die Anzahl der Zeilen- bzw. Spalteneinträge im Block $S_{t_1 t_2}$ beschränkt. In diesem Fall hat der Block $S_{t_1 t_2}$ beschränkten Rang.

Ist andererseits der Abstand größer Eins, $\text{dist}(t_1, t_2) > 1$, dann ist wegen Lemma 5.2 $(A_{t't'})_{t_1 t_2} = 0$. Daher ist

$$S_{t_1 t_2} = -A_{t_1 t} A_{tt}^{-1} A_{tt_2} = -\sum_{i,j \in t} A_{t_1 i} (A_{tt}^{-1})_{ij} A_{j t_2}.$$

Liegt i nicht in der in (5.7) definierten Nachbarschaft von t_1, dann ist $A_{t_1 i} = 0$. Falls $j \notin N(t_2)$, dann gilt $A_{j t_2} = 0$.

Mit der Bezeichnung $N^{\cap}(t_\ell) := N(t_\ell) \cap t$ gilt

$$S_{t_1 t_2} = -\sum_{i \in N^{\cap}(t_1),\, j \in N^{\cap}(t_2)} A_{t_1 i} (A_{tt}^{-1})_{ij} A_{j t_2}.$$

Da $t_1 \times t_2$ zulässig ist, gilt $t_1 \subset \mathcal{F}_\eta(t_2)$ oder $t_2 \subset \mathcal{F}_\eta(t_1)$. Aus dem Lemma 5.6 folgt dann $N(t_1) \subset \mathcal{F}_{2\eta}(N(t_2))$ oder $N(t_2) \subset \mathcal{F}_{2\eta}(N(t_1))$. Wegen (5.9) (wobei η durch 2η ersetzt wird) gibt es $U \in \mathbb{R}^{N^{\cap}(t_1) \times r}$ und $V \in \mathbb{R}^{N^{\cap}(t_2) \times r}$ mit $r \sim |\log \varepsilon|^d$, so dass

$$\left\|(A_{tt}^{-1})_{N^{\cap}(t_1) N^{\cap}(t_2)} - UV^T\right\|_2 < \varepsilon \left\|A_{tt}^{-1}\right\|_2. \tag{$*$}$$

Die Ergänzung von Nullzeilen bzw. -spalten in U und V führt auf $\hat{U} \in \mathbb{R}^{t\times r}$ und $\hat{V} \in \mathbb{R}^{t\times r}$. Mit dieser Notation gilt:

$$\begin{aligned} A_{t_1 t}\hat{U}\hat{V}^T A_{tt_2} &= \sum_{i\in N^\cap(t_1),\, j\in N^\cap(t_2)} \sum_{\ell=1}^{r} A_{t_1 i} U_{i\ell} V_{j\ell} A_{jt_2} \\ &= \sum_{\ell=1}^{r} \left(\sum_{i\in N^\cap(t_1)} A_{t_1 i} U_{i\ell} \right) \left(\sum_{j\in N^\cap(t_2)} V_{j\ell} A_{jt_2} \right) = XY^T, \end{aligned}$$

wobei $X \in \mathbb{R}^{t_1\times r}$, $Y \in \mathbb{R}^{t_2\times r}$ Matrizen mit den Spalten

$$X_{t_1\ell} := \sum_{i\in N^\cap(t_1)} A_{t_1 i} U_{i\ell} \quad \text{und} \quad Y_{t_2\ell} := \sum_{j\in N^\cap(t_2)} V_{j\ell} A_{jt_2}, \qquad \ell = 1, \ldots, r$$

sind.

Sei $B \in \mathbb{R}^{t\times t}$ eine Matrix mit den Einträgen

$$b_{ij} = \begin{cases} (A_{tt}^{-1})_{ij}, & \text{falls } i \in N^\cap(t_1) \text{ und } j \in N^\cap(t_2), \\ 0, & \text{sonst.} \end{cases}$$

Unter der Annahme (5.8) gilt

$$\begin{aligned} \left\| S_b - XY^T \right\|_2 = \left\| A_{t_1 t}(B - \hat{U}\hat{V}^T) A_{tt_2} \right\|_2 &\le \|A_{t_1 t}\|_2 \left\| (A_{tt}^{-1})_{N^\cap(t_1)N^\cap(t_2)} - UV^T \right\|_2 \|A_{tt_2}\|_2 \\ &\overset{(*)}{<} \varepsilon \|A_{t_1 t}\|_2 \left\| A_{tt}^{-1} \right\|_2 \|A_{tt_2}\|_2 \overset{\text{(5.8), Lemma 4.19}}{\le} c\,\varepsilon\, \mathrm{cond}_2(A) \|A\|_2 . \end{aligned} \tag{$**$}$$

Die Aussage des Satzes folgt mit Satz 4.20. □

5.2.2 $\mathcal{H}$-LU-Zerlegung

Ausgehend von der Blockstruktur

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} + \begin{bmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{bmatrix} = \begin{bmatrix} \tilde{L}_{11} & 0 \\ \tilde{L}_{21} & \tilde{L}_{22} \end{bmatrix} \begin{bmatrix} \tilde{U}_{11} & \tilde{U}_{12} \\ 0 & \tilde{U}_{22} \end{bmatrix}$$

sind folgende Probleme zu lösen:

$$\tilde{L}_{11}\tilde{U}_{11} = A_{11} + E_{11} \tag{5.10a}$$
$$\tilde{L}_{11}\tilde{U}_{12} = A_{12} + E_{12} \tag{5.10b}$$
$$\tilde{L}_{21}\tilde{U}_{11} = A_{21} + E_{21} \tag{5.10c}$$
$$\tilde{L}_{22}\tilde{U}_{22} = A_{22} + E_{22} - \tilde{L}_{21}\tilde{U}_{12}. \tag{5.10d}$$

Das Problem (5.10a) besitzt die gleiche Blockstruktur und kann rekursiv gelöst werden. Da die Vorwärtssubstitution (5.10b) und die Rückwärtssubstitution (5.10c) auf

Niedrigrang-Matrizen arbeiten, treten dabei die Fehler E_{12} und E_{21} auf. In der rekusiven Berechnung des approximativen Schur-Komplements (5.10d) ist daher nicht nur der Fehler E_{22} enthalten, sondern in (5.10d) wirken sich auch die Fehler E_{12} und E_{21} aus. Folglich sind im approximativen Schur-Komplement alle durch die Rekursion entstanden Approximationen enthalten, und eine Fehleranalyse nach [21] ist nicht möglich.

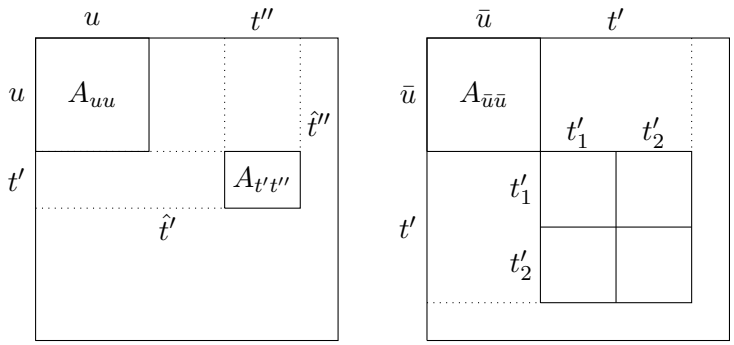
Daher wird im Folgenden zunächst eine rekursive Beziehung zwischen dem Schur-Komplement eines Blocks b und den Schur-Komplementen seiner Unterblöcke abgeleitet.

5.2.2.1 Eine Hierarchie von Schur-Komplementen

Seien $A \in \mathbb{R}^{I\times I}$, $t', t'' \subset I$ und $\hat{t}' = \{i \in I : i \leq \max t'\}$ sowie $\hat{t}'' = \{i \in I : i \leq \max t''\}$. Das Schur-Komplement des Blocks $t' \times t''$ ist

$$S(t', t'') = A_{t't''} - A_{t'u}A_{uu}^{-1}A_{ut''},$$

wobei $u := \{i \in I : i \leq \min t' \cup t''\} \neq \emptyset$, siehe Abb. 5.2a. Für $t' = t''$ ist $S(t', t'')$



(a) beliebiger Block (b) Diagonalblock

Abb. 5.2: Blockstruktur Schur-Komplement

das bekannte Schur-Komplement. Das folgende Lemma liefert eine Beziehung zwischen dem Schur-Komplement eines Diagonalblocks $t' \times t'$ und den Schur-Komplementen seiner Unterblöcke.

Lemma 5.8. *Seien t' ein Cluster des Clusterbaums T_I und t'_1, t'_2 seine Söhne. Das Schur-Komplement des Blocks $t' \times t'$ ist*

$$S(t', t') = \begin{bmatrix} S(t'_1, t'_1) & S(t'_1, t'_2) \\ S(t'_2, t'_1) & S(t'_2, t'_2) + S(t'_2t'_1)S(t'_1, t'_1)^{-1}S(t'_1, t'_2) \end{bmatrix}.$$

Beweis. Für eine geschlossene Darstellung der Theorie wird der Beweis aus [8] angegeben.

Das Schur-Komplement $S(t', t')$ habe die Blockstruktur

$$S(t', t') = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}.$$

Für die Blöcke $t'_1 \times t'_1$, $t'_1 \times t'_2$ und $t'_2 \times t'_1$ ist die Indexmenge u wie oben definiert. Die Schur-Komplemente dieser Blöcke sind

$$S(t'_1, t'_1) = A_{t'_1 t'_1} - A_{t'_1 u} A_{uu}^{-1} A_{u t'_1}, \qquad S(t'_1, t'_2) = A_{t'_1 t'_2} - A_{t'_1 u} A_{uu}^{-1} A_{u t'_2},$$
$$S(t'_2, t'_1) = A_{t'_2 t'_1} - A_{t'_2 u} A_{uu}^{-1} A_{u t'_1}.$$

Für den Block $t'_2 \times t'_2$ definieren wir $\bar{u} = \{i \in I : i \leq \min t'\}$, siehe Abb. 5.2b. Es gilt

$$S(t'_2, t'_2) = A_{t'_2 t'_2} - \begin{bmatrix} A_{t'_2 \bar{u}} & A_{t'_2 t'_1} \end{bmatrix} \begin{bmatrix} A_{\bar{u}\bar{u}} & A_{\bar{u} t'_1} \\ A_{t'_1 \bar{u}} & A_{t'_1 t'_1} \end{bmatrix}^{-1} \begin{bmatrix} A_{\bar{u} t'_2} \\ A_{t'_1 t'_2} \end{bmatrix}.$$

Wegen

$$\begin{aligned} \begin{bmatrix} A_{\bar{u}\bar{u}} & A_{\bar{u} t'_1} \\ A_{t'_1 \bar{u}} & A_{t'_1 t'_1} \end{bmatrix}^{-1} &= \begin{bmatrix} A_{\bar{u}\bar{u}}^{-1} + A_{\bar{u}\bar{u}}^{-1} A_{\bar{u} t'_1} S_{11}^{-1} A_{t'_1 \bar{u}} A_{\bar{u}\bar{u}}^{-1} & -A_{\bar{u}\bar{u}}^{-1} A_{\bar{u} t'_1} S_{11}^{-1} \\ -S_{11}^{-1} A_{t'_1 \bar{u}} A_{\bar{u}\bar{u}}^{-1} & S_{11}^{-1} \end{bmatrix} \\ &= \begin{bmatrix} A_{\bar{u}\bar{u}}^{-1} & -A_{\bar{u}\bar{u}}^{-1} A_{\bar{u} t'_1} S_{11}^{-1} \\ 0 & S_{11}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -A_{t'_1 \bar{u}} A_{\bar{u}\bar{u}}^{-1} & I \end{bmatrix} \end{aligned}$$

ist

$$\begin{aligned} S(t'_2, t'_2) &= A_{t'_2 t'_2} - \begin{bmatrix} A_{t'_2 \bar{u}} A_{\bar{u}\bar{u}}^{-1} & -A_{t'_2 \bar{u}} A_{\bar{u}\bar{u}}^{-1} A_{\bar{u} t'_1} S_{11}^{-1} + A_{t'_2 t'_1} S_{11}^{-1} \end{bmatrix} \begin{bmatrix} A_{\bar{u} t'_2} \\ S(t'_1, t'_2) \end{bmatrix} \\ &= A_{t'_2 t'_2} - \begin{bmatrix} A_{t'_2 \bar{u}} A_{\bar{u}\bar{u}}^{-1} & S(t'_2, t'_1) S_{11}^{-1} \end{bmatrix} \begin{bmatrix} A_{\bar{u} t'_2} \\ S_{12} \end{bmatrix} \\ &= A_{t'_2 t'_2} - A_{t'_2 \bar{u}} A_{\bar{u}\bar{u}}^{-1} A_{\bar{u} t'_2} - S_{21} S_{11}^{-1} S_{12} = S_{22} - S_{21} S_{11}^{-1} S_{12}. \end{aligned}$$

□

Jeder Block aus dem oberen Dreiecksteil der Matrix ist in einem Diagonalblock enthalten, siehe Abb. 5.3a. Aus dem Lemma 5.8 folgt

$$S(t', t'') = \begin{bmatrix} S(t'_1, t''_1) & S(t'_1, t''_2) \\ S(t'_2, t''_1) + S(t'_2, t'_1) S(t'_1, t'_1)^{-1} S(t'_1, t''_1) & S(t'_2, t''_2) + S(t'_2, t'_1) S(t'_1, t'_1)^{-1} S(t'_1, t''_2) \end{bmatrix}.$$

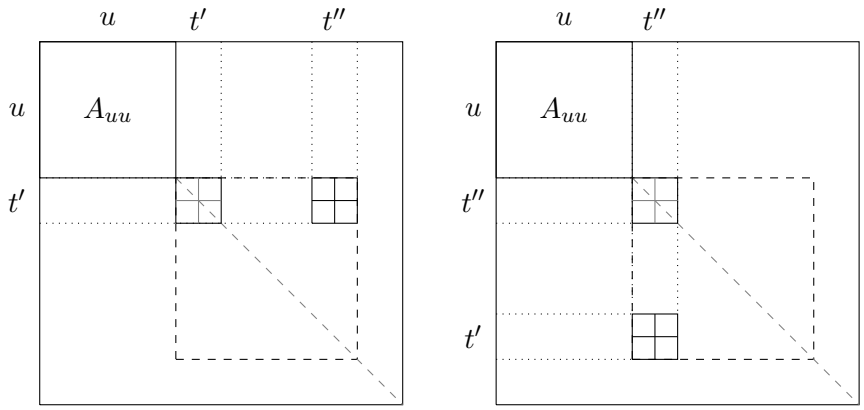
Befindet sich der Block $t' \times t''$ im unteren Dreiecksanteil der Matrix, siehe Abb. 5.3b, so erhält man mit dem Lemma 5.8 die folgende Beziehung:

$$S(t', t'') = \begin{bmatrix} S(t'_1, t''_1) & S(t'_1, t''_2) + S(t'_1, t''_1) S(t''_1, t''_1)^{-1} S(t''_1, t''_2) \\ S(t'_2, t''_1) & S(t'_2, t''_2) + S(t'_2, t''_1) S(t''_1, t''_1)^{-1} S(t''_1, t''_2) \end{bmatrix}.$$

5.2.2.2 Die Konstruktion der Faktoren $L_{\mathcal{H}}$ und $U_{\mathcal{H}}$

Zur Definition der symbolischen LU-Zerlegung sei das Schur-Komplement in der Blockstruktur des Lemmas 5.8 gegeben:

$$\begin{bmatrix} L_{11} & 0 \\ L_{21} & L_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix} = \begin{bmatrix} S(t_1, t_1) & S(t_1, t_2) \\ S(t_2, t_1) & S(t_2, t_2) + S(t_2 t_1) S(t_1, t_1)^{-1} S(t_1, t_2) \end{bmatrix}.$$



(a) Block im oberen Dreiecksanteil (b) Block im unteren Dreiecksanteil

Abb. 5.3: Schur-Komplement II

Mit $L_{11} := L(t_1)$ und $U_{11} := U(t_1)$ ergibt sich für die Außerdiagonalblöcke

$$\begin{aligned} L(t_1)U_{12} = S(t_1,t_2) \quad &\Rightarrow \quad U_{12} = L(t_1)^{-1}S(t_1,t_2), \\ L_{21}U(t_1) = S(t_2,t_1) \quad &\Rightarrow \quad L_{21} = S(t_2,t_1)U(t_1)^{-1}. \end{aligned}$$

Definiert man $L(t_2) := L_{22}$ und $U_{22} := U(t_2)$, dann gilt

$$S(t,t) = \begin{bmatrix} L(t_1) & 0 \\ S(t_2,t_1)U(t_1)^{-1} & L(t_2) \end{bmatrix} \begin{bmatrix} U(t_1) & L(t_1)^{-1}S(t_1,t_2) \\ 0 & U(t_2) \end{bmatrix}.$$

Nachdem wir nun die Blockstruktur der beteiligten Matrizen kennen, muss noch der Nachweis erbracht werden, dass die einzelnen Blöcke als hierarchische Matrix dargestellt werden können. Wir nutzen dazu die in den Abschnitten 5.1 und 5.2.1 angegebenen Voraussetzungen zur Darstellung des Schur-Komplements im hierarchischen Format.

Lemma 5.9. *Seien X und Y die Lösungen der Gleichungssysteme $L(t_1)X = S(t_1,t_2)$ sowie $YU(t_1) = S(t_2,t_1)$, wobei $\max t_1 \leq \min t_2$. Dann existieren für X und Y Näherungen $\tilde{X} \in \mathcal{H}(T_{t_1\times t_2}, r_S)$ sowie $\tilde{Y} \in \mathcal{H}(T_{t_2\times t_1}, r_S)$, so dass*

$$\left\|X - \tilde{X}\right\|_2 \leq c\,\mathrm{cond}_2(A)\log|I|\,\|U(t_1)\|_2 \quad \text{und} \quad \left\|Y - \tilde{Y}\right\|_2 \leq c\,\mathrm{cond}_2(A)\log|I|\,\|L(t_1)\|_2$$

mit dem in Satz 5.7 definierten blockweisen Rang r_S.

Beweis. Für die vollständige Darstellung geben wir den Beweis aus [8] an.

Wir zeigen zunächst, das ein $\tilde{X} \in \mathcal{H}(T_{t_1\times t_2}, r_s)$ zu X existiert, so dass auf jedem zulässigen Block

$$\left\|X_{t_1'\times t_2'} - \tilde{X}_{t_1'\times t_2'}\right\|_2 \leq c\ \mathrm{cond}_2(A)\,\varepsilon\ \left\|U(t_1')\right\|_2 \qquad (*)$$

gilt.

Sei $t_1 \times t_2 \in T_{I\times I}$ ein zulässiges Blatt. Nach Satz 5.7 kann das Schur-Komplement $S(t_1, t_2)$ durch eine Matrix $\tilde{S}(t_1, t_2) \in \mathbb{R}^{t_1\times t_2}$ mit Rang höchstens r_S approximiert werden. Folglich kann der Rang von $\tilde{X} := L(t_1)^{-1}\tilde{S}(t_1, t_2)$ nicht größer als r_S sein und es gilt

$$\begin{aligned}\left\|X - \tilde{X}\right\|_2 &= \left\|L(t_1)^{-1}\left[S(t_1,t_2) - \tilde{S}(t_1,t_2)\right]\right\|_2\\ &= \left\|U(t_1)S(t_1,t_1)^{-1}\left[S(t_1,t_2) - \tilde{S}(t_1,t_2)\right]\right\|_2\\ &\le \|U(t_1)\|_2 \left\|S(t_1,t_1)^{-1}\right\|_2 \left\|S(t_1,t_2) - \tilde{S}(t_1,t_2)\right\|_2\end{aligned}$$

Wegen $(**)$ im Beweis des Satzes 5.7 gilt:

$$\begin{aligned}&\le \varepsilon \left\|S(t_1,t_1)^{-1}\right\|_2 \|A\|_2 \|U(t_1)\|_2\\ &\overset{\text{Lemma 4.19}}{\le} \varepsilon \left\|A_{t_1t_1}^{-1}\right\|_2 \|A\|_2 \|U(t_1)\|_2\\ &\overset{(5.8)}{\le} c\,\varepsilon \left\|A^{-1}\right\|_2 \|A\|_2 \|U(t_1)\|_2 \le c\,\varepsilon\ \mathrm{cond}_2(A)\ \|U(t_1)\|_2 .\end{aligned}$$

Ist $t_1 \times t_2 \in T_{I\times I}$ kein Blatt, dann hat t_1 die Söhne t_1', t_1'' und t_2 die Söhne t_2', t_2''. Seien $X_{11} \in \mathbb{R}^{t_1'\times t_2'}$, $X_{12} \in \mathbb{R}^{t_1'\times t_2''}$, $X_{21} \in \mathbb{R}^{t_1''\times t_2'}$ und $X_{22} \in \mathbb{R}^{t_1''\times t_2''}$ durch

$$\begin{aligned} L(t_1')X_{11} &= S(t_1', t_2'), & L(t_1')X_{12} &= S(t_1', t_2'')\\ L(t_1'')X_{21} &= S(t_1'', t_2'), & L(t_1'')X_{22} &= S(t_1'', t_2'')\end{aligned}$$

definiert.

Per Induktion folgt, dass X_{11}, X_{12}, X_{21} und X_{22} durch $\mathcal{H}$-Matrizen $\tilde{X}_{11}$, $\tilde{X}_{12}$, $\tilde{X}_{21}$ und $\tilde{X}_{22}$, basierend auf den Unterbäumen von $T_{I\times I}$ mit den Wurzeln $t_1' \times t_2'$, $t_1' \times t_2''$, $t_1'' \times t_2'$ und $t_1'' \times t_2''$, approximiert werden können. Folglich erfüllt

$$X = \begin{bmatrix} X_{11} & X_{12}\\ X_{21} & X_{22}\end{bmatrix}$$

die Gleichung

$$\begin{aligned}L(t_1)X &= \begin{bmatrix} L(t_1') & 0\\ S(t_1'', t_1')U(t_1')^{-1} & L(t_1'')\end{bmatrix}\begin{bmatrix} X_{11} & X_{12}\\ X_{21} & X_{22}\end{bmatrix}\\ &= \begin{bmatrix} S(t_1', t_2)\\ S(t_1'', t_2) + S(t_1'', t_1')S(t_1', t_1')^{-1}S(t_1', t_2)\end{bmatrix}\\ &= S(t_1, t_2)\end{aligned}$$

und $\tilde{X}$ erfüllt die Forderung $(*)$.

Aus der Anwendung des Satzes 4.20 folgt die Aussage des Lemmas. Der Beweis für Y erfolgt analog. □

Der auf den Vorarbeiten der Abschnitte 5.1 und 5.2.1 und dieses Abschnitts aufbauende Satz liefert die Existenz der LU-Zerlegung.

Satz 5.10. *Sei P eine Partition von $I \times I$. Angenommen jedes Schur-Komplement $S(b)$, $b \in P$, einer Matrix A kann durch eine Matrix von Rang $r_S \sim (\log|I|)^\alpha |\log\varepsilon|^\beta$, mit einem Approximationsfehler ε und Konstanten $\alpha, \beta > 0$ dargestellt werden.*

Dann existieren eine untere und eine obere Dreiecksmatrix $L_\mathcal{H}, U_\mathcal{H} \in \mathcal{H}(P, r_S)$ und $c > 0$, so dass

$$\|A - L_\mathcal{H} U_\mathcal{H}\|_2 \leq c\,(\log|I|)\ \mathrm{cond}_2(A)\varepsilon\,\|L\|_2\,\|U\|_2\,,$$

wobei L und U die Faktoren der normalisierten LU-Zerlegung sind.

Beweis. Für die vollständige Darstellung geben wir den Beweis aus [8] an.

Es gilt $A = S(I,I) = L(I)U(I)$. Aufgrund der Eindeutigkeit der normalisierten LU-Zerlegung ist $L(I) = L$ und $U(I) = U$. Nach Lemma 5.9 gibt es hierarchische Matrizen $L_\mathcal{H}, U_\mathcal{H} \in \mathcal{H}(P, r_S)$, so dass

$$\|L - L_\mathcal{H}\|_2 \leq c\,\varepsilon\ \mathrm{cond}_2(A)\ \log(|I|)\ \|L\|_2 \quad \text{und} \quad \|U - U_\mathcal{H}\|_2 \leq c\,\varepsilon\ \mathrm{cond}_2(A)\ \log(|I|)\ \|U\|_2$$

gilt. Daher folgt

$$\begin{aligned}
\|A - L_\mathcal{H} U_\mathcal{H}\|_2 &\leq \|(L - L_\mathcal{H})U\|_2 + \|L(U - U_\mathcal{H})\|_2 + \|(L - L_\mathcal{H})(U - U_\mathcal{H})\|_2 \\
&\leq \|L - L_\mathcal{H}\|_2\,\|U\|_2 + \|L\|_2\,\|U - U_\mathcal{H}\|_2 + \|L - L_\mathcal{H}\|_2\,\|U - U_\mathcal{H}\|_2 \\
&\leq \Big(2\,c\,\varepsilon\ \mathrm{cond}_2(A)\ \log(|I|) + c^2\,\varepsilon^2\ \mathrm{cond}_2(A)^2\ (\log(|I|))^2\Big)\,\|L\|_2\,\|U\|_2\,.
\end{aligned}$$

□

5.3 Algebraische Konstruktion des Clusterbaums

Im Abschnitt 4.2.2 wurde der Clusterbaum unter Nutzung der geometrischen Informationen erstellt. Anstelle der Geometrie bildet nun der in (2.5) definierte Matrixgraph die Grundlage der Partitionierung.

Die $\mathcal{H}$-Inverse ist in der gleichen Zeitkomplexität zu berechnen wie die $\mathcal{H}$-LU-Zerlegung. Jedoch hat sich in der Praxis herausgestellt, dass die $\mathcal{H}$-LU-Zerlegung mit weniger Aufwand berechnet werden kann. Die $\mathcal{H}$-Inverse ist daher nur von theoretischem Interesse. Wir konzentrieren uns auf die Generierung eines Clusterbaums bzw. Block-Clusterbaums, der das Auffüllen der Faktoren L und U möglichst vermeidet.

Die im Abschnitt 3.2 über direkte Verfahren angeführten Umordnungsmethoden von Zeilen und Spalten, Cuthill-McKee, Reverse-Cuthill-McKee sowie Nested-Dissection, dienen der Reduktion des Auffüllens der Faktoren während der Dreieckszerlegung. In [47] werden Umordnungen bzgl. des minimalen Knotengrads mit Umordnungen durch Multilevel-Nested-Dissection im Hinblick auf die Qualität der Umordnungen, gemessen in der Anzahl der FLOPS[1] zur Faktorisierung, verglichen. Zum Vergleich nutzten die Autoren von [47] das Softwarepaket METIS[2], in welchem die Multilevel-Ideen von Hendrickson und Rothberg [43] eingesetzt werden. Dabei schneiden die Nested-Dissection-Ansätze im

[1] floating point operations

[2] `http://glaros.dtc.umn.edu/gkhome/metis/metis/overview`

Durchschnitt besser ab. Daher nutzen wir zur Konstruktion des Clusterbaums die von George vorgestellte *Teile-und-Herrsche* Strategie des *Nested-Dissection* [30].

Die Idee von Nested-Dissection besteht in der Zerlegung der Indexmenge I in die drei Teile t_1, s und t_2, wobei s die Mengen t_1 und t_2 separiert, und deshalb *Separator* genannt wird, siehe Abb. 5.4. Dabei soll die Zerlegung mit Nested-Dissection die Bedingungen

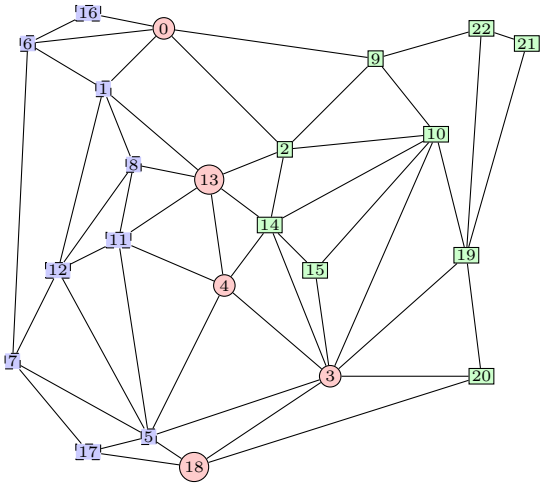


Abb. 5.4: Partitionierter Graph, $s = \{0, 13, 4, 3, 18\}$, $t_1 = \{16, 6, 1, 8, 12, 11, 7, 5, 17\}$ und $t_2 = \{22, 21, 9, 10, 2, 14, 19, 15, 20\}$

$$|t_1| \approx |t_2|, \tag{5.11a}$$

$$|s| \ll |t_1| \tag{5.11b}$$

erfüllen. Im Folgenden werden Index- bzw. Knotenmengen, welche Separatoren sind, meist s genannt. Ist die Index- bzw. Knotenmenge kein Separator oder ist es für den Zusammenhang nicht entscheidend, ob die Knotenmenge separiert, dann verwenden wir den Buchstaben t. Im Folgenden werden Cluster, welche keine Separatoren sind, auch als *reine* Cluster bezeichnet.

Nach Umordnung der Matrixeinträge ergibt sich die Matrixstruktur

$$\begin{bmatrix} A_{t_1t_1} & 0 & A_{t_1s} \\ 0 & A_{t_2t_2} & A_{t_2s} \\ A_{st_1} & A_{st_2} & A_{ss} \end{bmatrix}.$$

Während der LU-Zerlegung bleiben die Nullblöcke $A_{t_1t_2}$ und $A_{t_2t_1}$ erhalten, in den Blöcken A_{t_1s}, A_{t_2s}, A_{st_1}, A_{st_2} und A_{ss} werden jedoch neue Einträge generiert. Dies wird als fill-in bezeichnet. Zur Reduktion des fill-ins muss folglich der (Knoten-) Separator s möglichst klein sein, siehe Bedingung (5.11b).

In [46] wird die Dreiteilung mit Hilfe der geometrischen Information vorgenommen. Die im Folgenden vorgestellten Methoden nutzen nur die Matrix.

Die Zerlegung von I in t_1, s und t_2 wird in zwei Schritten realisiert. Im ersten Schritt wird der Matrixgraph in zwei etwa gleich große Teile t_1', t_2' zerlegt. Dabei entsteht der Kantenseparator $C := \{(i,j) : i \in t_1',\ j \in t_2'\} \subset E$. Im zweiten Schritt wird in dem bipartiten Graphen $B = (\partial t_1', \partial t_2', C)$, wobei $\partial t_1' := \{i \in t_1' : (i,j) \in C\}$ und $\partial t_2' := \{i \in t_2' : (i,j) \in C\}$ ein größtes Matching berechnet [45], um einen möglichst kleinen (Knoten-) Separator s zu erhalten. Zudem ist $t_1 := t_1' \backslash s$ sowie $t_2 := t_2' \backslash s$.

Solange die Mengen t_1 und t_2 noch nicht zu klein sind, werden die durch die Partitionierung der Knotenmenge erzeugten Teilgraphen $G_1 = (t_1, E(t_1))$ und $G_2 = (t_2, E(t_2))$, $E(t_k) := \{(i,j) \in E : i,j \in t_k\}$, $k = 1,2$, entsprechend rekursiv geteilt. Durch Anwendung des Alg. 16 entsteht der in Abb. 5.5 dargestellte Clusterbaum.

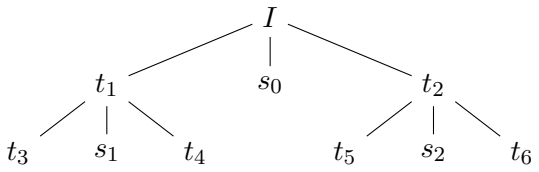


Abb. 5.5: Clusterbaum erzeugt mit Nested-Dissection-Methode

Algorithmus 16 Nested-Dissection ($G = (I, E)$)

1: teile I in die disjunkten Knotenmengen t_1', t_2' unter der Bedingung $|t_1'| \approx |t_2'|$
2: berechne minimale Knotenüberdeckung des bipartiten Graphen $B = (\partial t_1', \partial t_2', C)$
3: passe die Mengen t_1' und t_2' entsprechend an $\rightarrow$ t_1, s, t_2
4: **if** $|t_1| > n_{\min}$ **then**
5: Nested-Dissection ($G_1 = (t_1, E_1)$)
6: **if** $|t_2| > n_{\min}$ **then**
7: Nested-Dissection ($G_2 = (t_2, E_2)$)

Der erste Teil des Alg. 16 (Zeile 1) wird im Weiteren als *Bisektion* bezeichnet.

Aufgrund der rein algebraischen Clusterung können allgemeinere Gitter behandelt werden als mit der geometrischen Methode. Aus der Clusterung mit der Nested-Dissection Methode resultieren kardinalitätsbalancierte Clusterbäume, es existieren also Konstanten $c_b, C_b > 0$, so dass

$$c_b 2^{-\ell} |I| \le |t| \le C_b 2^{-\ell} |I| \tag{5.12}$$

für Cluster $t \in T_I^{(\ell)}$ gilt. Für FE-Gitter ist der Separator s eine Menge von niedrigerer Ordnung, d.h. es existieren Konstanten $c', C' > 0$, so dass

$$c' |t_1|^{1-1/d} \le |s| \le C' |t_1|^{1-1/d} \tag{5.13}$$

gilt, siehe [52]. Wir setzen die Äquivalenz von Durchmesser und Kardinalität voraus, d.h. es existieren Konstanten $c_1, c_2 > 0$, so dass

$$c_1 |t| \le (\mathrm{diam}(t))^d \le c_2 |t| \tag{5.14}$$

für alle $t \in T_I$ gilt. Die Cluster besitzen also ein „Volumen“.

Nach (5.12) gilt zum einen für $t \in T_I^{(\ell)}$: $\frac{1}{c_b|I|}|t|2^\ell \leq \frac{C_b}{c_b}$, zum anderen für $t' \in T_I^{(\ell')}$: $c_b|I| \leq |t'|2^{\ell'}$. Daraus folgt

$$\frac{|t|}{|t'|} \leq 2^{\ell'-\ell}\frac{C_b}{c_b}.$$

Wegen (5.14) ist $\frac{1}{c_2}(\operatorname{diam}(t))^d \leq |t|$ und $|t'| \leq \frac{1}{c_1}(\operatorname{diam}(t'))^d$. Insgesamt gilt daher

$$(\operatorname{diam}(t))^d \leq c_u 2^{\ell'-\ell}(\operatorname{diam}(t'))^d, \quad c_u := \frac{C_b c_2}{c_b c_1}. \tag{5.15}$$

5.3.1 Spectral-Nested-Dissection

Das Spectral-Nested-Dissection Verfahren basiert auf den Spektraleigenschaften des Matrixgraphen G. Pothen et al. [55] nutzen eine von Fiedler [27, 28] stammende Idee zur Teilung von G. Fiedler untersuchte die Beziehung zwischen dem zweitkleinsten Eigenwert der *Laplace-Matrix* und dem Kantenzusammenhang bzw. Knotenzusammenhang eines Graphen. Sei $M(G)$ die Adjazenzmatrix von G und

$$\mathcal{L}(G) := \operatorname{diag}(\deg v_1, \ldots, \deg v_n) - M(G) \tag{5.16}$$

seine Laplace-Matrix. Ausgehend von dem folgenden Satz betrachtete Fiedler die Beziehung zwischen dem Zusammenhang von Graphen und Eigenwerten bzw. Eigenvektoren der Laplace-Matrix.

Satz 5.11 (Fiedler, 1975). *Seien $G = (I, E)$ ein einfacher, ungerichteter und endlicher Graph bestehend aus n Knoten, $I = t_1' \cup t_2'$ eine Partition der Menge I sowie $\boldsymbol{x}$ der Vektor mit den Komponenten*

$$x_i = \begin{cases} 1, & \text{falls } i \in t_1', \\ -1, & \text{falls } i \in t_2'. \end{cases}$$

Dann ist die Größe des Kantenschnitts $C \subset E$ zwischen t_1' und t_2' gegeben durch

$$|C| = \frac{1}{4}\boldsymbol{x}^T \mathcal{L}(G)\boldsymbol{x}. \tag{5.17}$$

Beweis.

$$\begin{aligned}
\boldsymbol{x}^T \mathcal{L}(G)\boldsymbol{x} &= \sum_{i,j=1}^{n} \ell_{ij} x_i x_j = \sum_{i=1}^{n} \ell_{ii} x_i^2 + \sum_{i\neq j}^{n} \ell_{ij} x_i x_j \\
&= \sum_{i=1}^{n} \ell_{ii} + \sum_{\substack{(i,j)\in E \\ i,j\in t_1', i\neq j}} \ell_{ij} x_i x_j + \sum_{\substack{(i,j)\in E \\ i,j\in t_2', i\neq j}} \ell_{ij} x_i x_j + \sum_{\substack{(i,j)\in E \\ i\in t_1', j\in t_2'}} \ell_{ij} x_i x_j \\
&= \sum_{i=1}^{n} \deg v_i - \underbrace{\sum_{\substack{(i,j)\in E \\ i,j\in t_1', i\neq j}} 1}_{\text{Kanten in } t_1'} - \underbrace{\sum_{\substack{(i,j)\in E \\ i,j\in t_2', i\neq j}} 1}_{\text{Kanten in } t_2'} + \underbrace{\sum_{\substack{(i,j)\in E \\ i\in t_1', j\in t_2'}} 1}_{\text{Kanten zwischen } t_1' \text{ und } t_2'} \\
&= 2\cdot|E| - 2\cdot\left|E(t_1')\right| - 2\cdot\left|E(t_2')\right| + 2\cdot|C| = 4\cdot|C|
\end{aligned}$$

□

Um den fill-in während der LU-Faktorisierung zu reduzieren, soll der (Knoten-) Separator möglichst klein sein (siehe Bedingung (5.11b)). Weil dieser auf der Basis des Kantenseparators berechnet wird, soll $|C|$ möglichst klein sein. Daher wird (5.17) als ein diskretes Optimierungsproblem mit den Nebenbedingungen

$$\begin{aligned}
&\boldsymbol{x}^T\boldsymbol{x} = n,\ x_i \in \{-1, 1\} \quad \text{sowie} \\
&\boldsymbol{x}^T\mathbf{1} = 0, \quad \mathbf{1} = (1,\ldots,1)^T \quad \text{(entspricht Forderung (5.11b))}
\end{aligned}$$

aufgefasst. Dieses Optimierungsproblem ist NP-vollständig, siehe [63]. Ein Ausweg besteht in der Lösung des kontinuierlichen Optimierungsproblems

$$f(\boldsymbol{z}) = \boldsymbol{z}^T \mathcal{L}(G)\boldsymbol{z} \ \to \min \tag{5.18}$$

mit den Nebenbedingungen

$$\boldsymbol{z}^T\boldsymbol{z} = n,\ \boldsymbol{z} \in \mathbb{R}^n \quad \text{und} \quad \boldsymbol{z}^T\mathbf{1} = 0.$$

Wegen $\mathcal{L}(G)\cdot\mathbf{1} = 0$ ist der Vektor mit konstanten Einträgen ein Eigenvektor zum Eigenwert $\lambda_1 = 0$. Außerdem ist die Laplace-Matrix von G symmetrisch und positiv semidefinit. Sie besitzt daher nur reelle Eigenwerte und eine Basis $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n \in \mathbb{R}^n$ aus orthonormalen Eigenvektoren. Die Darstellung von $\boldsymbol{z} \in \mathbb{R}^n$ in dieser Basis ist

$$\boldsymbol{z} = \sum_{\ell=1}^{n} c_\ell \boldsymbol{u}_\ell$$

mit Koeffizienten $c_\ell \in \mathbb{R}$. Wegen

$$0 = \boldsymbol{z}^T\mathbf{1} = \langle \boldsymbol{u}_1 | \boldsymbol{z} \rangle = \sum_{\ell=1}^{n} c_\ell \langle \boldsymbol{u}_1 | \boldsymbol{u}_\ell \rangle$$

ist $c_1 = 0$. Setzen wir $\boldsymbol{z}$, dargestellt in der Eigenvektorbasis, in (5.18) ein und nutzen aus, dass

$$n = \boldsymbol{z}^T \boldsymbol{z} = \langle \boldsymbol{z} | \boldsymbol{z} \rangle = \sum_{\ell,k=1}^{n} c_\ell c_k \langle \boldsymbol{u}_\ell | \boldsymbol{u}_k \rangle = \sum_{\ell=2}^{n} c_\ell^2,$$

so erhalten wir

$$\boldsymbol{z}^T \mathcal{L}(G) \boldsymbol{z} = \left(\sum_{\ell=2}^{n} c_\ell \boldsymbol{u}_\ell \right)^T \left(\sum_{\ell=2}^{n} c_\ell \lambda_\ell \boldsymbol{u}_\ell \right) = \sum_{\ell=2}^{n} c_\ell^2 \lambda_\ell \geq \lambda_2 \sum_{\ell=2}^{n} c_\ell^2 = \lambda_2 n.$$

Die Partitionierung anhand des zu λ_2 gehörigen Eigenvektors $\boldsymbol{u}_2$ liefert (heuristisch) den minimalen Kantenseparator. Dieser besteht nach (5.17) aus mindestens $\frac{1}{4}\lambda_2 n$ Kanten.

Algorithmus 17 Spectral-Bisection

1: berechne den zweitkleinsten Eigenwert λ_2 von $\mathcal{L}(G)$
2: bestimme Median m des zu λ_2 gehörenden Eigenvektors $\boldsymbol{u}_2$
3: teile die Knotenmenge gemäß $t_1' := \{i \in I : \boldsymbol{u}_2(i) \leq m\}, \quad t_2' := I \backslash t_1'$

Da die Berechnung von Eigenwerten großer Matrizen numerisch sehr aufwändig ist, wird Spectral-Bisection in Kombination mit Multilevel-Methoden eingesetzt.

5.3.2 Nested-Dissection basierend auf Multilevel-Zugang

Die Partitionierung der Knoten des Matrixgraphen (2.5) in t_1, s und t_2 umfasst im Alg. 16 zwei Schritte. Der erste Schritt besteht aus einer Bisektion der Knotenmenge. Im vorigen Abschnitt wurde die Zerlegung auf Basis der numerisch aufwändigen Berechnung des zweitkleinsten Eigenwerts/Eigenvektors vorgestellt. Die spektrale Bisektion wird durch die Bisektion aus METIS ersetzt, siehe Abb. 5.6.

Phase 1: Schrittweises Vergröbern des Graphen. Sei $G = G^{(0)}$. Wir vergröbern $G^{(0)}$, indem wir ein maximales Matching $M^{(0)}$ in $G^{(0)}$ berechnen und die Endknoten einer Kante $m = (i_m, j_m) \in M^{(0)}$ zu einem neuen Knoten z_m zusammenfassen. Die mit den Knoten i_m und j_m in $G^{(0)}$ inzidenten Kanten werden mit dem Knoten z_m inzident. Wir erhalten einen bzgl. der Kardinalität der Knotenmenge kleineren Graphen $G^{(1)}$. Der eben beschriebene Algorithmus wird solange wiederholt angewendet, bis der Graph nur noch wenige Knoten enthält. Es ergibt sich eine Sequenz von Graphen $G^{(0)}, G^{(1)}, \ldots, G^{(m)}$ mit $|V^{(0)}| \geq |V^{(1)}| \geq \cdots \geq |V^{(m)}|$. In [47] werden verschiedene Matching-Strategien untersucht. Wir nutzen das sogenannte *heavy edge matching (HEM)*, da dabei die Knoten, welche durch Kanten mit besonders großem Gewicht verbunden sind, ausgewählt werden. Folglich wird das Gesamtkantengewicht des Graphen reduziert. Dies führt meist zu einem kleinen Kantenseparator.

Phase 2: Bisektion des gröbsten Graphen. Da nur ein relativ kleiner Graph geteilt werden muss, können sowohl kombinatorische Methoden, wie beispielsweise der KL-Algorithmus [48], als auch spektrale Bisektion eingesetzt werden. Der Aufwand für die Teilung von $G^{(m)}$ ist relativ gering, verglichen mit dem Aufwand $G^{(0)}$ zu teilen.

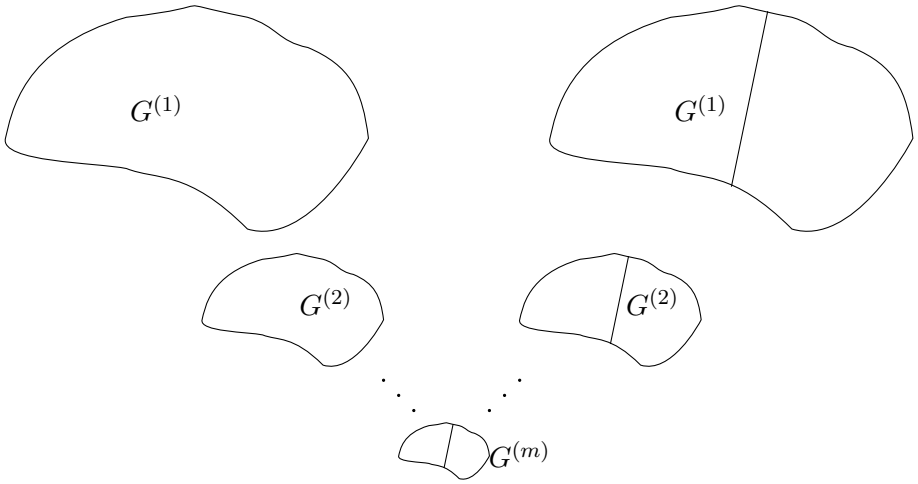


Abb. 5.6: Multilevel Graphbisektion

Phase 3: Schrittweises Verfeinern des Graphen. Die Partitionierung von $G^{(m)}$ wird auf den Graphen $G^{(m-1)}$ übertragen, die Partitionierung von $G^{(m-1)}$ auf $G^{(m-2)}$ usw. Wir erhalten schließlich eine Zweiteilung von G.

Sowohl bei der spektralen Bisektion als auch bei der Multilevel Bisektion wird sicher gestellt, dass $|t_1'| \approx |t_2'|$.

5.3.3 Erweiterter Nested-Dissection-Clusterbaum

Für unsere Konstruktion des Block-Clusterbaums ist es wichtig, dass jede Ebene des Clusterbaums eine vollständige Partition von I enthält und dass die (Teil-)Indexmengen in einer Stufe etwa die gleiche Größe besitzen. Unter Berücksichtigung von (5.12) wird daher s in die nächste Ebene kopiert, solange die Bedingung

$$c_b 2^{-\ell'} |I| \leq |s| \leq C_b 2^{-\ell'} |I| \tag{5.19}$$

noch *nicht* erfüllt ist. Wir erhalten eine Sequenz $s^{(1)}, \ldots, s^{(\ell'-\ell)}$ von Separatoren, die die gleiche Indexmenge enthalten, jedoch zu unterschiedlichen Ebenen des Clusterbaums gehören. Wir bezeichnen diese Knoten als *virtuelle Knoten*. Virtuelle Knoten entstehen also nicht unmittelbar aus der Teilung eines Clusters. Ist (5.19) erfüllt, wird s rekursiv in zwei Teile partitioniert, siehe Abb. 5.7. Um eine linear-logarithmische Komplexität der Matrixoperationen zu erhalten, darf der Clusterbaum nur aus einer in $|I|$ linearen Anzahl von Knoten bestehen. Für den in Definition 4.7 erklärten Clusterbaum gilt Lemma 4.8. Außerdem gilt:

Lemma 5.12. *Der erweiterte Nested-Dissection-Clusterbaum Algorithmus fügt $\mathcal{O}(|I|)$ (virtuelle) Knoten in T_I ein. Damit ist eine lineare Gesamtanzahl von Knoten in T_I bzgl. $|I|$ garantiert.*

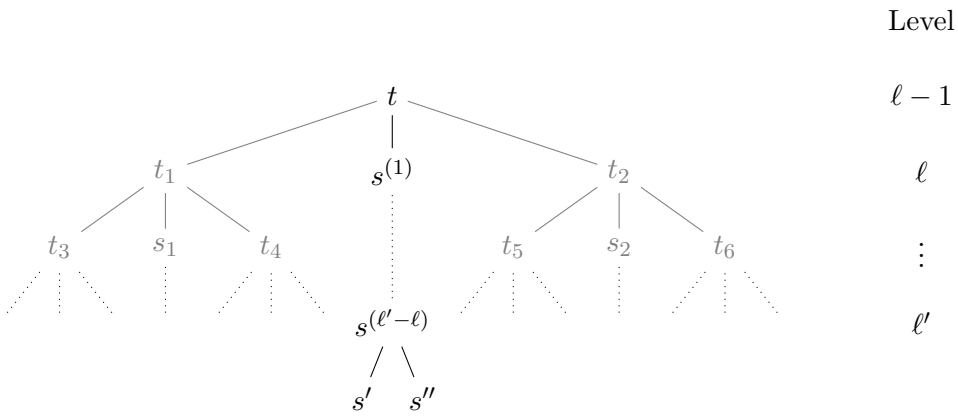


Abb. 5.7: erweiterter Clusterbaum erzeugt mit Nested-Dissection-Methode

Beweis. Sei $s \in T_I^{(\ell)}$ ein Separator. Die Indexmenge s wird in die nächste Ebene kopiert, solange die Bedingung (5.19) noch *nicht* erfüllt ist. Sei ℓ' die Ebene, in welcher $|s|$ die Bedingung (5.19) erfüllt, siehe Abb. 5.7. Mit (5.13), (5.12) und (5.19) folgt, dass

$$c' \left(c_b |I| 2^{-\ell} \right)^{1-1/d} \leq |s| < C_b |I| 2^{-\ell'} \quad \Longrightarrow \quad \ell' - \ell < \frac{1}{d} \left(\log_2 |I| - \ell \right) + k_1,$$

gilt, wobei $k_1 := \log_2 \frac{C_b}{c' c_b^{1-1/d}}$.

Falls $\ell \geq \ell_{\max} := d(k_1 - 1) + \log_2 |I|$ ist, dann ist der Ebenenunterschied $\ell' - \ell$ kleiner als Eins, und aus dem Separator entstehen keine virtuellen Knoten. Daher müssen nur die Ebenen $\ell \leq \ell_{\max}$ betrachtet werden.

Die Anzahl der virtuellen Knoten in T_I ergibt sich aus der Summation über alle Ebenen $\ell = 1, \ldots, \ell_{\max}$ in T_I. In der ℓ-ten Ebene von T_I gibt es höchstens $2^{\ell-1}$ Separatoren. Jeder Separator ist Startpunkt eines Pfads, bestehend aus virtuellen Knoten, höchstens der Länge $\frac{1}{d} \left(\log_2 |I| - \ell \right) + k_1 = \frac{1}{d} \left(\ell_{\max} + d - \ell \right)$. Die Anzahl der virtuellen Knoten ist daher beschränkt durch

$$\begin{aligned}
\sum_{\ell=1}^{\ell_{\max}} \frac{2^{\ell-1}}{d} \left(\ell_{\max} + d - \ell \right) &= \frac{1}{d} \left((\ell_{\max} + d) \sum_{\ell=1}^{\ell_{\max}} 2^{\ell-1} - \sum_{\ell=1}^{\ell_{\max}} \ell 2^{\ell-1} \right) \\
&= \frac{1}{d} \left((\ell_{\max} + d) \left(2^{\ell_{\max}} - 1 \right) - \sum_{\ell=1}^{\ell_{\max}} \ell 2^{\ell-1} \right) \\
&= \frac{1}{d} \left((\ell_{\max} + d + 1) \left(2^{\ell_{\max}} - 1 \right) - \ell_{\max} 2^{\ell_{\max}} \right) \\
&\leq \left(1 + \frac{1}{d} \right) 2^{\ell_{\max}} = \left(1 + \frac{1}{d} \right) 2^{d(k_1-1)} |I|,
\end{aligned}$$

wobei wir die Beziehung

$$\sum_{k=1}^{n} k 2^{k-1} = 1 + (n-1) 2^n.$$

verwendet haben. □

Beispiel 5.13. Die Anwendung des Multilevel-Nested-Dissection Algorithmus auf die in Abb. 4.1 dargestellte Geometrie führt zu dem in Abb. 5.8 skizzierten Clusterbaum. Im Gegensatz zu den mit geometrischen Methoden erzeugten Clusterbäumen aus Abb. 4.2 bzw. Abb. 4.4 ist dieser *balanciert*.

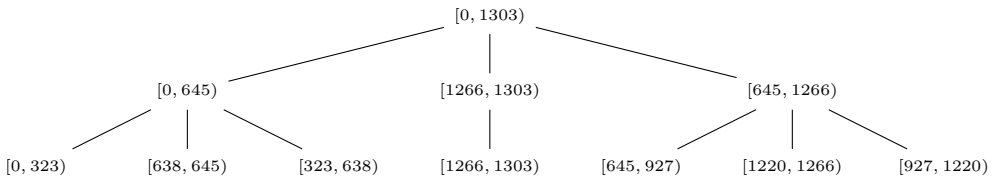


Abb. 5.8: Drei Stufen des durch Zerlegung mit der Multilevel-Nested-Dissection Methode generierten Clusterbaums für Abb. 4.1

5.3.4 Numerische Ergebnisse

Die spektrale Clusterung wird anhand des folgenden Testbeispiels mit der Multilevel-Clusterung verglichen.

Beispiel 5.14. Gegeben sei die partielle Differentialgleichung $\mathcal{L}\,u = f$ mit dem in (1.3) definierten Operator $\mathcal{L}$ auf dem drei-dimensionalen Gebiet der Abb. 5.9. Auf dem Rand seien Null-Randbedingungen vorgegeben. Die Koeffizienten der Matrix $C(x)$, $x \in \Omega$, wer-

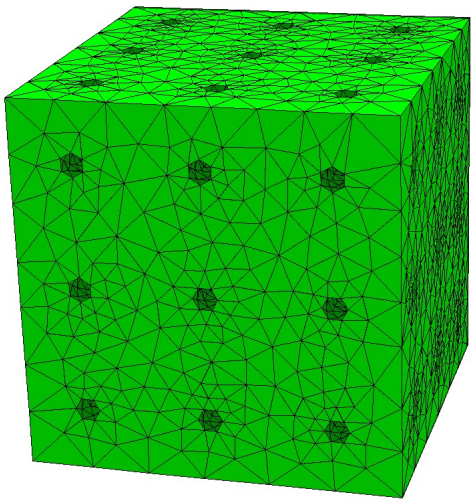


Abb. 5.9: Berechnungsgebiet

den zufällig aus dem Intervall $[0, 1000]$ so gewählt, dass die Matrix C positiv definit ist. Zur Diskretisierung werden lineare finite Elemente genutzt. Die resultierenden linearen

Gleichungssysteme ergeben sich aus unterschiedlich feinen FE-Gittern. Die FE-Gitter wurden mit dem Programm NETGEN[3] erzeugt und sind zu den Löchern hin adaptiv verfeinert. NETGEN bietet fünf Stufen zur Erzeugung von Gittern: *very coarse*, *coarse*, *moderate*, *fine* und *very fine*. Die generierten Gitter können jeweils uniform verfeinert werden. Die im Folgenden verwendeten Matrizen sind in Tab. 5.1 beschrieben.

ID	Ausgangsgitter	Verfeinerungs-schritte	Matrixgröße n	Nicht-Null-Einträge nnz	$\frac{nnz}{n}$
F	fine	0	101 296	1 368 594	13.51
M1	moderate	1	297 927	4 134 255	13.88
VF	very fine	0	658 609	9 294 721	14.11
C2	coarse	2	906 882	12 854 824	14.17
M2	moderate	2	2 539 954	36 768 808	14.48
VF1	very fine	1	5 413 520	78 935 174	14.58

Tabelle 5.1: Testmatrizen

Die Anzahl der Cluster eines mit Spectral-Nested-Dissection konstruierten Clusterbaums unterscheidet sich nicht wesentlich von der Anzahl eines mit Multilevel-Nested-Dissection berechneten Clusterbaums. Wie die Werte in Tab. 5.2 zeigen, ist der Multilevel-Algorithmus jedoch wesentlich schneller als die spektrale Clusterung. Daher wird im Folgenden die Multilevel-Clusterung eingesetzt.

ID	Spectral-Nested-Dissection		Multilevel-Nested-Dissection	
	Anzahl der Cluster	Zeit in s	Anzahl der Cluster	Zeit in s
F	4 455	5.93	4 446	1.76
M1	9 093	22.32	8 989	5.71
VF	28 108	71.15	27 821	14.30
C2	36 993	90.51	36 581	21.62
M2	115 091	350.84	113 125	69.38
VF1	233 087	991.96	230 813	168.10

Tabelle 5.2: Spektrale und Multilevel-Clusterung für minimale Blockgröße $n_{\min} = 50$

5.4 Algebraische Konstruktion des Block-Clusterbaums

Sei der Cluster $t \in T_I^{(\ell)}$ in t_1, s und t_2 durch den in Abschnitt 5.3 beschriebenen algebraischen Algorithmus zerlegt. Die (Knoten- bzw. Index-)Menge s separiert dann die Mengen t_1 und t_2, d.h. es existiert keine Kante zwischen einem $i \in t_1$ und $j \in t_2$, siehe Abb. 5.4. Daher gilt nach Definition des Matrixgraphen, dass $a_{ij} = 0$ und $a_{ji} = 0$. Die Matrix besitzt demnach die (Block-)Struktur

$$\begin{bmatrix} A_{t_1 t_1} & 0 & A_{t_1 s} \\ 0 & A_{t_2 t_2} & A_{t_2 s} \\ A_{s t_1} & A_{s t_2} & A_{ss} \end{bmatrix}. \tag{5.20}$$

[3] `http://www.hpfem.jku.at/netgen/` bzw. `http://sourceforge.net/projects/netgen-mesher/`

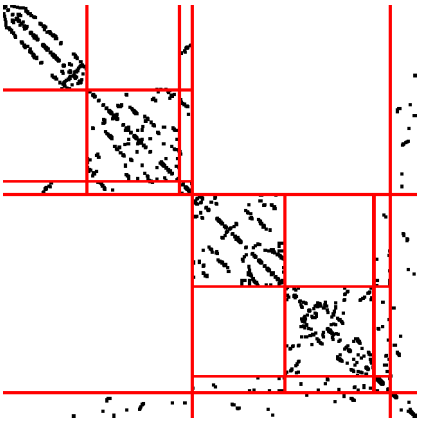


Abb. 5.10: Matrix Struktur nach zwei Nested-Dissection Schritten

Die rekursive Anwendung des Partitionierungsalgorithmus ergibt die in Abb. 5.10 dargestellte Struktur der Matrixeinträge. Daher wird die Sohnabbildung (4.14) modifiziert, d.h.

$$S_{I\times I}(t_1\times t_2) := \begin{cases} \emptyset, & \text{falls } S_I(t_1)=\emptyset \text{ oder } S_I(t_2)=\emptyset,\\ \emptyset, & \text{falls } t_1\neq t_2 \text{ und weder } t_1 \text{ noch } t_2 \text{ Separator},\\ \emptyset, & \text{falls } t_1\times t_2 \text{ zulässig},\\ S_I(t_1)\times S_I(t_2), & \text{sonst.}\end{cases} \tag{5.21}$$

Für die Komplexitätsangaben des Abschnitts 4.3 wird die Beschränktheit von c_{sp} unabhängig von $|I|$ benutzt. Diese Bedingung muss auch durch die algebraische Matrixpartition erfüllt werden.

Lemma 5.15. *Der Block-Clusterbaum sei unter Beachtung der algebraischen Zulässigkeitsbedingung* (5.4) *konstruiert. Unter der Annahme, dass* (5.1), (5.12) *und* (5.14) *erfüllt sind, gilt*

$$c_{\mathrm{sp}} \le 2c_v c_2 \frac{C_b}{c_b}\left(2+\frac{1}{\eta}\right)^d.$$

Beweis. Seien t_1 ein Cluster aus der ℓ-ten Ebene des Clusterbaums, $i_0\in t_1$ und

$$N_\rho := \{t_2\in T_I^{(\ell)} : \max_{j\in t_2} d(i_0,j)\le\rho\}$$

die Nachbarschaft, $\rho>0$. Wegen

$$|N_\rho|\, c_b\, |I|\, 2^{-\ell} \overset{(5.12)}{\le} \sum_{t_2\in N_\rho} |t_2| \overset{(5.1)}{\le} c_v\,\rho^d, \tag{5.22}$$

enthält N_ρ von t_1 höchstens $\frac{c_v}{c_b}\,\rho^d\,\frac{2^\ell}{|I|}$ Cluster t_2 der gleichen Ebene des Clusterbaums.

Wir wählen ein $t_2 \in T_I$ derart, dass $t_1 \times t_2 \in T_{I\times I}$. Seien t_1^* und t_2^* Väter von t_1 und t_2. Angenommen es gilt $\max_{j\in t_2} d(i_0, j) \geq \rho_0$, wobei

$$\rho_0 := \frac{1}{\eta}\min\{\operatorname{diam}(t_1^*), \operatorname{diam}(t_2^*)\} + \operatorname{diam}(t_1^*) + \operatorname{diam}(t_2^*) \overset{(5.14)}{\leq} (c_2 C_b |I| 2^{-(\ell-1)})^{1/d}\left(2 + \frac{1}{\eta}\right)$$

ist. Dann folgt aus

$$\operatorname{dist}(t_1^*, t_2^*) \geq \max_{j\in t_2} d(i_0, j) - \operatorname{diam}(t_1^*) - \operatorname{diam}(t_2^*) \geq \frac{1}{\eta}\min\{\operatorname{diam}(t_1^*), \operatorname{diam}(t_2^*)\},$$

dass $t_1^* \times t_2^*$ zulässig ist. Daher kann $t_1 \times t_2$ nicht in $T_{I\times I}$ sein. Folglich muss die Annahme falsch gewesen sein, und es gilt

$$\max_{j\in t_2} d(i_0, j) < \rho_0 \leq (c_2 C_b |I| 2^{-(\ell-1)})^{1/d}\left(2 + \frac{1}{\eta}\right).$$

Wegen $\{t_2 \in T_I : t_1 \times t_2 \in T_{I\times I}\} \subset N_{\rho_0}$ erhalten wir aus (5.22):

$$\max_{t_1\in T_I} |\{t_2 \in T_I : t_1 \times t_2 \in T_{I\times I}\}| \leq 2c_v c_2 \frac{C_b}{c_b}\left(2 + \frac{1}{\eta}\right)^d.$$

Die Rollen von t_1 und t_2 können o.B.d.A. vertauscht werden, und man erhält eine Abschätzung für $\max_{t_2\in T_I} |\{t_1 \in T_I : t_1 \times t_2 \in T_{I\times I}\}|$ in analoger Weise. □

Die im Abschnitt 4.3.5.2 eingeführte Größe c_{id} wird für Abschätzungen des Rangs während der Multiplikation benutzt. Auch für die algebraische Partitionierung muss c_{id} unabhängig von der Problemgröße beschränkt sein.

Lemma 5.16. *Unter der Annahme, dass Durchmesser und Größe eines Clusters die Bedingung* (5.14) *erfüllen, gilt*

$$c_{\mathrm{id}} \leq \left[c_u^{1/d}(2 + \eta)\right]^{3d}.$$

Beweis. Sei $b \in \mathcal{L}(T_{I\times I})$ aus der ℓ-ten Ebene des Block-Clusterbaums. Ist b ein nicht zulässiges Blatt, dann ist $c_{\mathrm{id}}(b) = 1$. Für zulässige Blätter $b = t_1 \times t_2$ definieren wir

$$q := d \log_2 c_u(2 + \eta).$$

Wir werden zeigen, dass für Cluster $t_1', r', t_2' \in T_I$, $t_1' \times t_2' \subset b = t_1 \times t_2$, die $t_1' \times r', r' \times t_2' \in T_{I\times I}^{(\ell+q)}$ erfüllen, folgt, dass $t_1' \times r'$ oder $r' \times t_2'$ Blätter im Block-Clusterbaum $T_{I\times I}$ sind. Das folgt aus

$$\begin{aligned}
\operatorname{diam}(r') &= (1 + \eta/2)\operatorname{diam}(r') - \eta/2 \operatorname{diam}(r') \\
&\overset{(5.15)}{\leq} (1 + \eta/2)c_u^{1/d} 2^{-q/d} \min\{\operatorname{diam}(t_1), \operatorname{diam}(t_2)\} - \eta/2 \operatorname{diam}(r') \\
&\leq \frac{1}{2}\min\{\operatorname{diam}(t_1), \operatorname{diam}(t_2)\} - \eta/2 \operatorname{diam}(r') \\
&\leq \frac{\eta}{2}(\operatorname{dist}(t_1, t_2) - \operatorname{diam}(r')) \\
&\leq \eta \max\{\operatorname{dist}(t_1', r'), \operatorname{dist}(r', t_2')\}.
\end{aligned}$$

Daher ist entweder $t_1' \times r'$ oder $r' \times t_2'$ zulässig. Einer der beiden Blöcke hat folglich keine Nachfolger in $T_{I\times I}$. Deshalb ist die Anzahl der im Block b enthaltenen Subblöcke beschränkt durch $c_{\mathrm{id}} \leq 7^q \leq [c_u^{1/d}(2+\eta)]^{3d}$. □

Zur Auswertung der Zulässigkeitsbedingung sind der Abstand zwischen den Indexmengen t_1 und t_2 sowie der jeweilige Durchmesser zu bestimmen. Für $\mathrm{dist}(t_1, t_2)$ sind $\mathcal{O}(|t_1| \cdot |t_2|)$ Abstandsberechnungen durchzuführen. Hinzu kommen für $\mathrm{diam}(t_1)$ etwa $|t_1|^2$ Abstandsberechnungen innerhalb von t_1 bzw. für $\mathrm{diam}(t_2)$ etwa $|t_2|^2$ Abstandsberechnungen innerhalb von t_2.

Obwohl im geometrischen Fall Abstände zwischen zwei Punkten in $\mathcal{O}(1)$ berechnet werden können, müssen bei der Auswertung der geometrischen Zulässigkeitsbedingung (4.13) die Abstände zwischen Clustern approximativ bestimmt werden, um eine linear-logarithmische Laufzeit zur Konstruktion des Block-Clusterbaums zu erreichen.

Im algebraischen Fall wird der Abstand zwischen zwei Knoten durch Breitensuche im Matrixgraphen $G = (I, E)$ mit $\mathcal{O}(|I|)$ Operationen berechnet. Falls die linear-logarithmische Gesamtkomplexität nicht überschritten werden soll, können Abstände bzw. Durchmesser von Clustern nicht exakt bestimmt werden,

Für den Durchmesser wird in Abschnitt 5.5 eine Approximation vorgestellt. Das Abstandsproblem wird in Abschnitt 5.6 behandelt.

5.5 Durchmesserapproximation

Eine Näherung für den Durchmesser eines Clusters t bzw. seines zugeordneten Graphen G kann mit der Breitensuche während der Clusterung bestimmt werden. Dazu wird Alg. 1 modifiziert, indem zusätzlich ein Knoten mit maximalem Abstand zurück geliefert wird. Eine erste Näherung für den Durchmesser ist der maximale Abstand zwischen einem (gewählten) Startknoten i und einem Knoten j, der $j = \max_{\nu\in V(G)} d(i,\nu)$ erfüllt. Diese Näherung kann durch k-fache Anwendung der Breitensuche verbessert werden, $k \in \mathbb{N}$ klein, siehe Alg. 18.

Algorithmus 18 Berechnung des Durchmessers: $\widetilde{\mathrm{diam}}\,(G)$

```
1: wähle Startpunkt i ∈ V(G)
2: d_max := 0
3: for ℓ = 0, ..., k do
4:     j := BFS (G, i)
5:     if d_max < d(i, j) then
6:         d_max := d(i, j)
7:     i := j
```

Für die Approximation von $\widetilde{\mathrm{diam}}(t)$ gilt

$$r(t) \leq \widetilde{\mathrm{diam}}(t) \leq \mathrm{diam}(t) \leq 2r(t), \tag{5.23}$$

wobei $r(t)$ der im Kapitel 2 erklärte Radius des Graphen t ist.

Die bei der Clusterung der Indexmenge der Testmatrix mit 101 296 Zeilen bzw. Spalten aus Tab. 5.1 berechneten approximativen Durchmesser werden mit den exakten Durchmessern verglichen. Tab. 5.3 listet die prozentuale Verteilung der Verhältnisse $\widetilde{\operatorname{diam}} \,/\, \operatorname{diam}$ auf die Intervalle auf. Die Zahlen belegen, dass mit der approximativen Variante der Durchmesserberechnung im Durchschnitt sehr gute Ergebnisse erzielt werden.

Intervall	Anteil in Prozent
$[0.65, 0.70)$	0.11
$[0.70, 0.75)$	0.16
$[0.75, 0.80)$	0.89
$[0.80, 0.85)$	1.18
$[0.85, 0.90)$	1.14
$[0.90, 0.95)$	0.16
$[0.95, 1.00]$	95.67

Tabelle 5.3: prozentuale Verteilung der Verhältnisse $\widetilde{\operatorname{diam}} \,/\, \operatorname{diam}$ für $k = 3$

5.6 Abstandsberechnung

5.6.1 Mengenschnitt

Es sei

$$\mathcal{F}_\eta(t_1) := \{t \in T_I^{(\ell)} : \operatorname{diam}(t_1) \leq \eta \operatorname{dist}(t_1, t)\}$$

das *Fernfeld* von $t_1 \in T_I^{(\ell)}$ und $\mathcal{N}_\eta(t_1) := \{t \in T_I^{(\ell)} : t \notin \mathcal{F}_\eta(t_1)\}$ das *Nahfeld* von t_1. Mit diesen Begriffen lässt sich die Zulässigkeitsbedingung (5.4) umformulieren: Ein Block $t_1 \times t_2$ ist genau dann nicht zulässig, wenn $t_2 \in \mathcal{N}_\eta(t_1)$ und $t_1 \in \mathcal{N}_\eta(t_2)$.

Die in [34] vorgeschlagene Methode zur approximativen Auswertung basiert auf dieser alternativen Formulierung. Für die Bestimmung der Zulässigkeit eines Blocks ist nicht der genaue Wert des Abstands zwischen t_1 und t_2 wichtig, sondern ob $t_2 \in \mathcal{N}_\eta(t_1)$ oder $t_1 \in \mathcal{N}_\eta(t_2)$ sind. Zum Test der Bedingung $t_1 \in \mathcal{N}_\eta(t_2)$ werden die ρ-Umgebungen

$$U_\rho(t_2) := \{j \in V(G) : i \in t_2 : d(i, j) \leq \rho\},$$

$\rho = 1, \ldots, \lceil \frac{1}{\eta} \min\{\operatorname{diam} t_1, \operatorname{diam} t_2\} \rceil$, schrittweise konstruiert. Nach jedem Vergrößerungsschritt wird der Schnitt von t_1 und der Umgebung $U_\rho(t_2)$ gebildet. Ist der Schnitt für $\rho < \lceil \frac{1}{\eta} \min\{\operatorname{diam} t_1, \operatorname{diam} t_2\} \rceil$ nicht leer, dann ist der Abstand der Cluster kleiner als für die Zulässigkeit benötigt. Bleibt der Schnitt für $\rho = \lceil \frac{1}{\eta} \min\{\operatorname{diam} t_1, \operatorname{diam} t_2\} \rceil$ leer, so ist $t_1 \times t_2$ zulässig.

Für einen Zulässigkeitstest werden $\mathcal{O}\left(d^d \max\{|t_1|, |t_2|\}\right)$ Operationen benötigt, siehe [34].

Die mit der Mengenschnitt-Methode konstruierte Partition wird im Weiteren als *Mengenschnitt-Partition* bezeichnet.

5.6.2 Konstruktion eines Clustergraphen

Wir werden im Folgenden eine Methode zur approximativen Abstandsberechnung vorstellen, die eine von der Größe der Cluster unabhängige Anzahl von Operationen benötigt.

Definition 5.17. Die (disjunkten) Indexmengen $t_1, t_2 \subset I$ heißen *benachbart*, falls im Matrixgraphen eine Kante von t_1 nach t_2 existiert. D.h. t_1 ist benachbart zu t_2, falls

$$\exists i \in t_1,\, \exists j \in t_2 : \quad (i,j) \in E.$$

Die Nachbarschaften können während der Konstruktion des Clusterbaums berechnet werden. Da benachbarte Cluster den Abstand Eins haben, kann die Zulässigkeit eines Blocks unmittelbar durch die Abfrage der Durchmesser angegeben werden. Sind $t_1, t_2 \in T_I^{(\ell)}$ benachbart und $\operatorname{diam}(t_1) \leq \eta$ oder $\operatorname{diam}(t_2) \leq \eta$, dann ist $t_1 \times t_2$ zulässig, weil

$$\min\{\operatorname{diam}(t_1), \operatorname{diam}(t_2)\} \leq \eta \cdot 1 = \eta \operatorname{dist}(t_1, t_2)$$

gilt.

Aus der Partitionierung von t mit Nested-Dissection in t_1, s und t_2 folgt, dass t_1 mit s und t_2 mit s benachbart sind. Da zwischen t_1 und t_2 keine Kante existiert, sind t_1 und t_2 nicht benachbart. Wie auch aus der Blockstruktur der Matrix (5.20) erkennbar ist, besteht der Block $t_1 \times t_2$ folglich nur aus Nullen. Deshalb ist eine Berechnung des Abstands zwischen t_1 und t_2 unnötig.

Die Bedingung (5.4) muss daher nur dann ausgewertet werden, falls die Cluster nicht benachbart sind und einer der beiden Cluster ein Separator ist.

Seien $t_1, t_2 \in T_I^{(\ell)}$ zwei nicht benachbarte Cluster. Zur Approximation des Abstands wird ein sogenannter *Clustergraph* G_D konstruiert, dessen Knotenmenge aus Clustern besteht, und dessen Kantenmenge durch die Nachbarschaft der Cluster induziert wird. Im Clusterbaum existieren benachbarte Cluster t_1^* und t_2^*, wobei $t_1 \subseteq t_1^*$, $t_2 \subseteq t_2^*$ und t_1^*, t_2^* unter allen benachbarten Clustern, welche t_1 oder t_2 enthalten, maximale Tiefe haben. Die Knotenmenge von G_D besteht aus den Clustern der ℓ-ten Ebene, deren Weg zur Wurzel über t_1^* oder t_2^* führt. Zwischen zwei Knoten existiert eine Kante, falls die den Knoten zugeordneten Indexmengen benachbart sind. Alg. 19 fasst die Schritte zusammen.

Algorithmus 19 Konstruiere G_D (t_1, t_2)

1: bestimme $t_1^*, t_2^* \in T_I$
2: bestimme $V(G_D) = \{\hat{t} \in T_I^{(\ell)} : \hat{t} \subseteq t_1^* \text{ oder } \hat{t} \subseteq t_2^*\}$
3: bestimme $E(G_D) = \{(\hat{t}, \check{t}) : \hat{t}, \check{t} \in V(G_D), \hat{t}, \check{t} \text{ benachbart}\}$

Beispiel 5.18. In Abb. 5.11 sind die sich aus der Konstruktion des Clusterbaums ergebenden Nachbarschaften durch gestrichelte Linien gekennzeichnet. Die berechneten Nachbarschaften werden durch gepunktete Linien angedeutet. Es soll der näherungsweise Abstand zwischen t_7 und s_0'' berechnet werden. Aus $t_1^* = t_3$ und $t_2^* = s_0'$ ergibt sich die Knotenmenge $\{t_7, s_3, t_8, s_0''\}$. In Abb. 5.12 wird der zugehörige Graph G_D dargestellt.

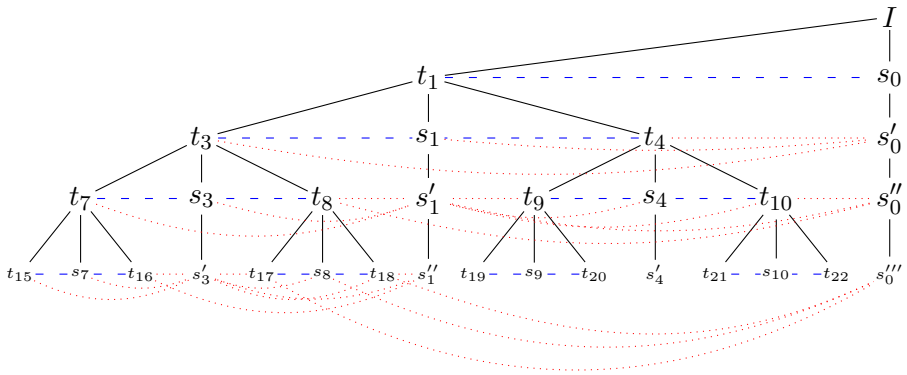


Abb. 5.11: Teil eines Clusterbaums mit durch gestrichelte (a priori bekannt) oder gepunktete (müssen berechnet werden) Linien symbolisierten Nachbarschaften

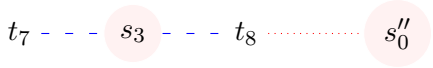


Abb. 5.12: G_D zur Berechnung einer Näherung für $\operatorname{dist}(t_7, s''_0)$.

Beispiel 5.19. Sei der Block $t_7 \times s''_0$ nicht zulässig. Dann muss u. a. die Zulässigkeit des Blocks $t_{15} \times s'''_0$ untersucht werden (Abb. 5.11). Wegen $t_1^* = t_3$ und $t_2^* = s'_0$ ergibt sich die Knotenmenge $\{t_{15}, s_7, t_{16}, s'_3, t_{17}, s_8, t_{18}, s'''_0\}$ für G_D. Insgesamt ergibt sich der Graph aus Abb. 5.13.

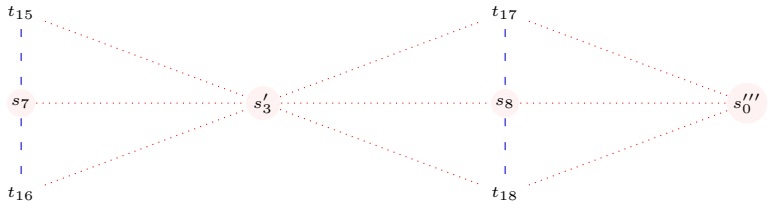


Abb. 5.13: G_D zur Berechnung einer Näherung für $\operatorname{dist}(t_{15}, s'''_0)$.

Um die linear-logarithmische Komplexität zur Konstruktion einer $\mathcal{H}$-Matrix nicht zu überschreiten, ist eine möglichst effiziente Auswertung der Zulässigkeitsbedingung notwendig. D.h. die Konstruktion darf höchstens $\mathcal{O}(|I|)$ Operationen benötigen. Das folgende Lemma zeigt, dass G_D in der Größe beschränkt ist, und daher Abstände in einer von $|I|$ unabhängigen Komplexität berechnet werden können.

Da zwischen nicht benachbarten Clustern $w_1, w_2 \in T_I^{(\ell)}$ mindestens ein Cluster der gleichen Stufe liegt, wird die Bedingung (5.24) im folgenden Lemma vorausgesetzt.

Lemma 5.20. *Seien $t_1 \times t_2 \in T_{I\times I}^{(\ell)}$, wobei t_1, t_2 nicht benachbart sind. Wir nehmen an,*

dass eine Konstante $c > 0$ existiert, so dass für jedes Paar von nicht benachbarten Clustern w_1, w_2 der gleichen Stufe in T_I gilt

$$\operatorname{dist}(w_1, w_2) \geq c \min\{\operatorname{diam}(w_1), \operatorname{diam}(w_2)\}. \tag{5.24}$$

Ist (5.14) *erfüllt, dann ist die Kardinalität der Knotenmenge von G_D durch eine Konstante beschränkt.*

Beweis. Seien $p_1, p_2 \in T_I^{(\ell^*)}$ die benachbarten Vorfahren von t_1, t_2 mit maximaler Tiefe im Clusterbaum.

Seien q_1 und q_2 die nicht benachbarten Söhne von p_1 und p_2, wobei $t_1 \subset q_1$ und $t_2 \subset q_2$ und seien f_1, f_2 die Väter von t_1 und t_2. Da f_1, f_2 Nachfolger von q_1, q_2 sind, gilt

$$\begin{aligned}\operatorname{dist}(f_1, f_2) &\geq \operatorname{dist}(q_1, q_2) \geq c \min\{\operatorname{diam}(q_1), \operatorname{diam}(q_2)\} \\ &\overset{(5.14)}{\geq} c\, c_1^{1/d} \min\{|q_1|^{1/d}, |q_2|^{1/d}\} = c\, c_1^{1/d} |I|^{1/d} 2^{-\frac{\ell^*+1}{d}}.\end{aligned}$$

Da $f_1 \times f_2$ als nicht zulässig vorausgesetzt wird (denn sonst würde der Block $t_1 \times t_2$ nicht existieren), gilt

$$\eta\, c\, c_1^{1/d} |I|^{1/d} 2^{-\frac{\ell^*+1}{d}} \leq \eta \operatorname{dist}(f_1, f_2) < \min\{\operatorname{diam}(f_1), \operatorname{diam}(f_2)\} \overset{(5.14)}{\leq} c_2^{1/d} |I|^{1/d} 2^{-(\ell-1)/d},$$

was zu der Beziehung

$$\frac{1}{4}\frac{c_1}{c_2}(c\eta)^d < 2^{-(\ell-\ell^*)} \quad \Longleftrightarrow \quad \ell - \ell^* < 2 + \log_2 \frac{c_2}{c_1}(c\eta)^{-d}$$

führt.

Daher besteht der Graph G_D, welcher zur approximierten Abstandsberechnung zwischen den Clustern t_1 und t_2 konstruiert wird, aus einer durch $2 \cdot 3^{\ell-\ell^*}$ begrenzten Anzahl von Knoten. □

Sei t ein Cluster im Matrixgraphen und p ein kürzester Weg zwischen t_1 und t_2 der durch t führt. Der in t verlaufende Anteil des Weges wird mit $p_{|t}$ bezeichnet, seine Länge durch $|p_{|t}|$ symbolisiert.

Bemerkung 5.21. Jeder Knoten in G_D repräsentiert einen Teilgraphen des Matrixgraphen G_A. Es sei $p_{G_D}(t_1, t_2) = (t_1 = u_0, u_1, \ldots, u_k = t_2)$ ein mit Breitensuche bestimmter kürzester Weg der Länge k in G_D. Dabei treten die folgenden beiden Probleme auf:

1. Es muss kein kürzester Weg zwischen t_1 und t_2 in G_A (vollständig) innerhalb der Teilgraphen u_j, $j = 1, \ldots, k$, liegen.

2. Pro Knoten $u_j \in V(G_D)$, $j = 1, \ldots, k$, über den der Weg von t_1 nach t_2 führt, wird die Weglänge genau um Eins erhöht. Ein kürzester Weg in G_A könnte genau einen Knoten des Teilgraphen u_j oder aber wesentlich mehr als einen Knoten enthalten, genauer gilt $1 \leq |p_{|u_j}| \leq \operatorname{diam}(u_j)$. Der Wegverlauf innerhalb der Teilgraphen wird im obigen Algorithmus nicht berücksichtigt.

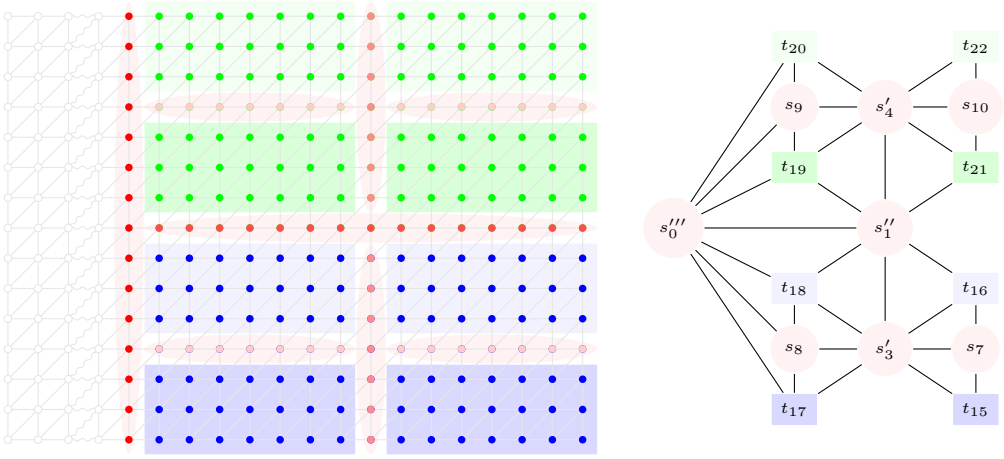


Abb. 5.14: Konstruktion von G_D aus dem Matrixgraphen

Beispiel 5.22. Auf der linken Seite von Abb. 5.14 ist ein Ausschnitt eines Matrixgraphen skizziert, rechts daneben der Graph G_D, wie er etwa bei der Zulässigkeitsabfrage der Cluster s'''_0 und t_{16} entsteht. Ein kürzester Weg von s'''_0 nach t_{16} hat im Matrixgraphen die Länge neun und führt durch die Cluster t_{18} und s'_3. In G_D ist $p_{G_D}(s'''_0, t_{16}) = (s'''_0, s''_1, t_{16})$, d.h. es ergibt sich eine Weglänge von zwei.

Wie das Beispiel verdeutlicht, können große Diskrepanzen zwischen exaktem und approximiertem Abstand entstehen. Ungenaue Abstandsapproximationen können in zu feinen oder zu groben Partitionen resultieren. Wird die Partition zu fein, d.h. es werden mehr Blöcke generiert als nötig, erhöhen sich die Laufzeit der Algorithmen und der Speicherbedarf. Eine zu feine Partition kann durch die im Abschnitt 4.1.7 beschriebene Zusammenfassung von Blöcken vergröbert werden. Andererseits muss in einer zu groben Partition der blockweise Rang erhöht werden, um die Genauigkeit zu garantieren.

In einer Analyse der Berechnung der Zulässigkeitsbedingungen muss beachtet werden, dass eine Diskrepanz zwischen exaktem und approximiertem Abstand nicht immer eine falsche Zulässigkeitsentscheidung liefert. In der folgenden Bemerkung sind die verschiedenen Fälle aufgeschlüsselt.

Bemerkung 5.23. **1** Der Block $t_1 \times t_2$ ist zulässig, d.h. es gilt $\frac{1}{\eta} \min\{\operatorname{diam}(t_1), \operatorname{diam}(t_2)\} \leq \operatorname{dist}(t_1, t_2)$.

- **1.1** Ist $\operatorname{dist}(t_1, t_2) \leq \widetilde{\operatorname{dist}}(t_1, t_2)$, wird der Block korrekt als zulässig erkannt.
- **1.2** Ist $\operatorname{dist}(t_1, t_2) > \widetilde{\operatorname{dist}}(t_1, t_2)$, wird der Block
  - **1.2.1** im Fall von $\frac{1}{\eta} \min\{\operatorname{diam}(t_1), \operatorname{diam}(t_2)\} \leq \widetilde{\operatorname{dist}}(t_1, t_2) < \operatorname{dist}(t_1, t_2)$ korrekt als zulässig erkannt.
  - **1.2.2** im Fall von $\widetilde{\operatorname{dist}}(t_1, t_2) < \frac{1}{\eta} \min\{\operatorname{diam}(t_1), \operatorname{diam}(t_2)\} \leq \operatorname{dist}(t_1, t_2)$ unnötigerweise verfeinert.

2 Der Block $t_1 \times t_2$ ist unzulässig, d.h. es gilt $\operatorname{dist}(t_1, t_2) < \frac{1}{\eta} \min\{\operatorname{diam}(t_1), \operatorname{diam}(t_2)\}$.

2.1 Ist $\operatorname{dist}(t_1,t_2) < \widetilde{\operatorname{dist}}(t_1,t_2)$, wird der Block

2.1.1 im Fall von $\widetilde{\operatorname{dist}}(t_1,t_2) < \operatorname{dist}(t_1,t_2) < \frac{1}{\eta}\min\{\operatorname{diam}(t_1),\operatorname{diam}(t_2)\}$ korrekt als unzulässig erkannt.

2.1.2 im Fall von $\operatorname{dist}(t_1,t_2) < \frac{1}{\eta}\min\{\operatorname{diam}(t_1),\operatorname{diam}(t_2)\} \leq \widetilde{\operatorname{dist}}(t_1,t_2)$ zu groß und hat einen zu großen Rang.

2.2 Ist $\operatorname{dist}(t_1,t_2) \geq \widetilde{\operatorname{dist}}(t_1,t_2)$, wird der Block korrekt als unzulässig erkannt.

Da die Zulässigkeit eines Blocks $t_1 \times t_2$ durch das Verhältnis zwischen dem minimalen Durchmesser und Abstand gegeben ist, wird der Quotient

$$q(t_1,t_2) = \frac{\widetilde{\operatorname{dist}}(t_1,t_2)}{\operatorname{dist}(t_1,t_2)} \tag{5.25}$$

betrachtet. Im Idealfall ist für jeden Block $q(t_1,t_2) = 1$. In der Praxis liegen die Quotienten im Intervall $[q_{\min}, q_{\max}]$.

Entsprechend der Fallunterscheidung der Bemerkung 5.23 wird die Partition

$$[q_{\min}, q_{\max}] = [q_{\min}, \frac{1}{\mu}) \cup [\frac{1}{\mu}, \mu) \cup [\mu, q_{\max}] \tag{5.26}$$

gewählt. Das Intervall $[q_{\min}, \frac{1}{\mu})$ soll dabei dem Fall 1.2.2 entsprechen, das Intervall $[\frac{1}{\mu}, \mu)$ die Fälle 1.1, 1.2.1, 2.1.1 und 2.2 beinhalten und das Intervall $[\mu, q_{\max}]$ den Fall 2.1.2 abdecken.

Lediglich im Fall 2.1.2 muss der Rang erhöht werden, um die geforderte blockweise Genauigkeit zu garantieren. Im Beweis der Aussage $\operatorname{rank}[p(A)_{t_1 t_2}] \leq c_v(1+\eta)^d k^d$ (erster Teil des Lemmas 5.2) wird ohne Einschränkung $k \geq \operatorname{dist}(t_1,t_2)$ angenommen. Dann gilt

$$k \geq \operatorname{dist}(t_1,t_2) = \frac{\widetilde{\operatorname{dist}}(t_1,t_2)}{q(t_1,t_2)},$$

und es folgt $r'(q) \sim q^d r$. Die Kosten eines Blocks $t_1 \times t_2$ sind dann proportional zu

$$\min\{r'(q(t_1,t_2))(|t_1|+|t_2|), |t_1|\cdot|t_2|\}.$$

Um die linear-logarithmische Komplexität in $|I|$ zu garantieren, darf der Abstand nur für wenige, kleine Blöcke überschätzt werden. In dem Ausdruck

$$\alpha(q) = \begin{cases} \displaystyle\sum_{q(t_1,t_2)\in[q,q+1)} \min\{q^d r(|t_1|+|t_2|), |t_1|\cdot|t_2|\}, & q > 1, \\ \displaystyle\sum_{q(t_1,t_2)\in\left[\frac{1}{q+1},\frac{1}{q}\right)} r(|t_1|+|t_2|), & q \leq 1, \end{cases} \tag{5.27}$$

gehen sowohl das Verhältnis (5.25) als auch die Blockgröße ein. Der Ausdruck muss für $q \to \infty$ exponentiell abfallen. In den numerischen Experimenten stellt sich heraus, dass q beschränkt ist.

Zur Verbesserung der Abstandsapproximation werden im Folgenden verschiedene Methoden vorgestellt. Da wir für die Abstandsapproximation keine zu (5.23) ähnliche Abschätzung haben, wird die Annahme

$$\widetilde{\operatorname{dist}}(t_1, t_2) \approx \operatorname{dist}(t_1, t_2), \tag{5.28}$$

wobei $\widetilde{\operatorname{dist}}(t_1, t_2)$ durch eines der im Folgenden vorgestellten Verfahren gegeben ist, numerisch überprüft.

5.6.2.1 Kantengewichte per Größe der Nachbarschaft

Das zweite in der Bemerkung 5.21 beschriebene Problem soll durch die Wahl von Kantengewichten abgemindert werden. Unter der Annahme $|p_{|t}| \sim \operatorname{diam} t$ wird der Abstand i.A. zu groß, unter der Annahme $|p_{|t}| = 1$ meist zu klein.

Es sei $N(t_1, t_2) := \{i \in t_1 : (i, j) \in E(G_A), j \in t_2\}$ die Nachbarschaft von t_1 mit t_2, $|N(t_1, t_2)|$ sei die Größe der Nachbarschaft. Ist $N(t_1, t_2)$ groß, so existieren im Matrixgraphen viele Kanten zwischen den Clustern t_1 und t_2. Folglich gibt es auch viele verschiedene Wege von t_1 in t_2 bzw. umgekehrt, welche über die Nachbarschaft führen. Die Länge des Anteils des Weges durch den Cluster ist daher möglicherweise kürzer als dessen Durchmesser. Daher wird der Kante $(t_1, t_2) \in E(G_D)$ durch die Gewichtsfunktion

$$w(t_1, t_2) = \frac{\min\{\widetilde{\operatorname{diam}}\, t_1, \widetilde{\operatorname{diam}}\, t_2\}}{|N(t_1, t_2)|}$$

ein geringes Gewicht zugeordnet. Gibt es wenige Kanten zwischen t_1 und t_2 im Matrixgraphen, dann wird der Kante (t_1, t_2) ein großes Gewicht zugeordnet. Der Clustergraph mit dieser Gewichtung wird mit G_D^N bezeichnet.

Beispielsweise entsteht aus der in Abb. 5.14 angedeuteten Clusterung des Matrixgraphen der Graph der Abb. 5.15. Da G_D^N ein gewichteter Graph ist, wird statt der Breitensuche der Algorithmus von Dijkstra zur Abstandsbestimmung eingesetzt.

Um die Asymptotik zu untersuchen, wird zu jeder FE-Matrix aus Tab. 5.1 ein Block-Clusterbaum mittels der beschriebenen Abstandsapproximation erstellt. Für alle nach der approximativen Zulässigkeitsbedingung zulässigen Blöcke $t_1 \times t_2$ wird zusätzlich der exakte Abstand zwischen t_1 und t_2 berechnet und der Quotient (5.25) gebildet. Für

Matrix-größe	$\lvert\{q \in [q_{\min}, 0.5)\}\rvert$ in %	$\lvert\{q \in [0.5, 2.0)\}\rvert$ in %	$\lvert\{q \in [2.0, q_{\max}]\}\rvert$ in %
101 296	0.50	37.13	62.37
297 927	1.37	48.74	49.89
658 609	2.97	49.74	47.29
906 882	2.74	48.24	49.03
2 539 954	4.54	54.65	40.80
5 413 520	4.78	53.47	41.75

Tabelle 5.4: Anteil der Quotienten in den Intervallen $[q_{\min}, 0.5)$, $[0.5, 2.0]$ sowie $[2.0, q_{\max}]$ für die Abstandsberechnung in G_D^N

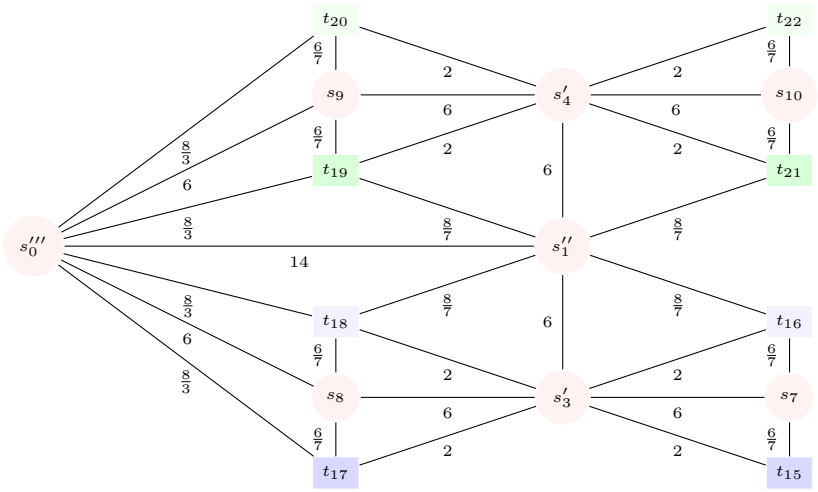


Abb. 5.15: G_D mit Kantengewichten nach Größe der Nachbarschaft

die Resultate in der Tab. 5.4 wird die Intervallpartition (5.26) betrachtet, wobei $\mu = 2$ gewählt wurde. Die Werte in der zweiten Spalte sind im Vergleich zur dritten und vierten Spalte klein. Die Partition ist daher nur in wenigen Blöcken zu fein. Aus der letzten Spalte erkennt man, dass die Methode meist zu große Werte für den Abstand liefert und die Partition folglich zu grob wird. Um die Genauigkeit in den zu großen Blöcken zu garantieren, muss der Rang in diesen erhöht werden. Die grobe Partition spiegelt sich auch in den Funktionsverläufen der Funktion (5.27) in den Abb. 5.16a bis Abb. 5.16f wider. In den Abbildungen ist insbesondere das Intervall $[\mu, q_{\max}]$ detaillierter als in der Tab. 5.4 aufgeschlüsselt. Die Funktion $\alpha(q)$ fällt tendenziell für größer werdendes q. Das Verhältnis zwischen approximiertem Abstand und exaktem Abstand scheint beschränkt zu sein. In der Praxis wird der Block für große q aber im vollen Format gespeichert und damit der Technik der hierarchischen Matrizen eine wesentliche Basis entzogen. Diese Wahl der Gewichte wird daher verworfen.

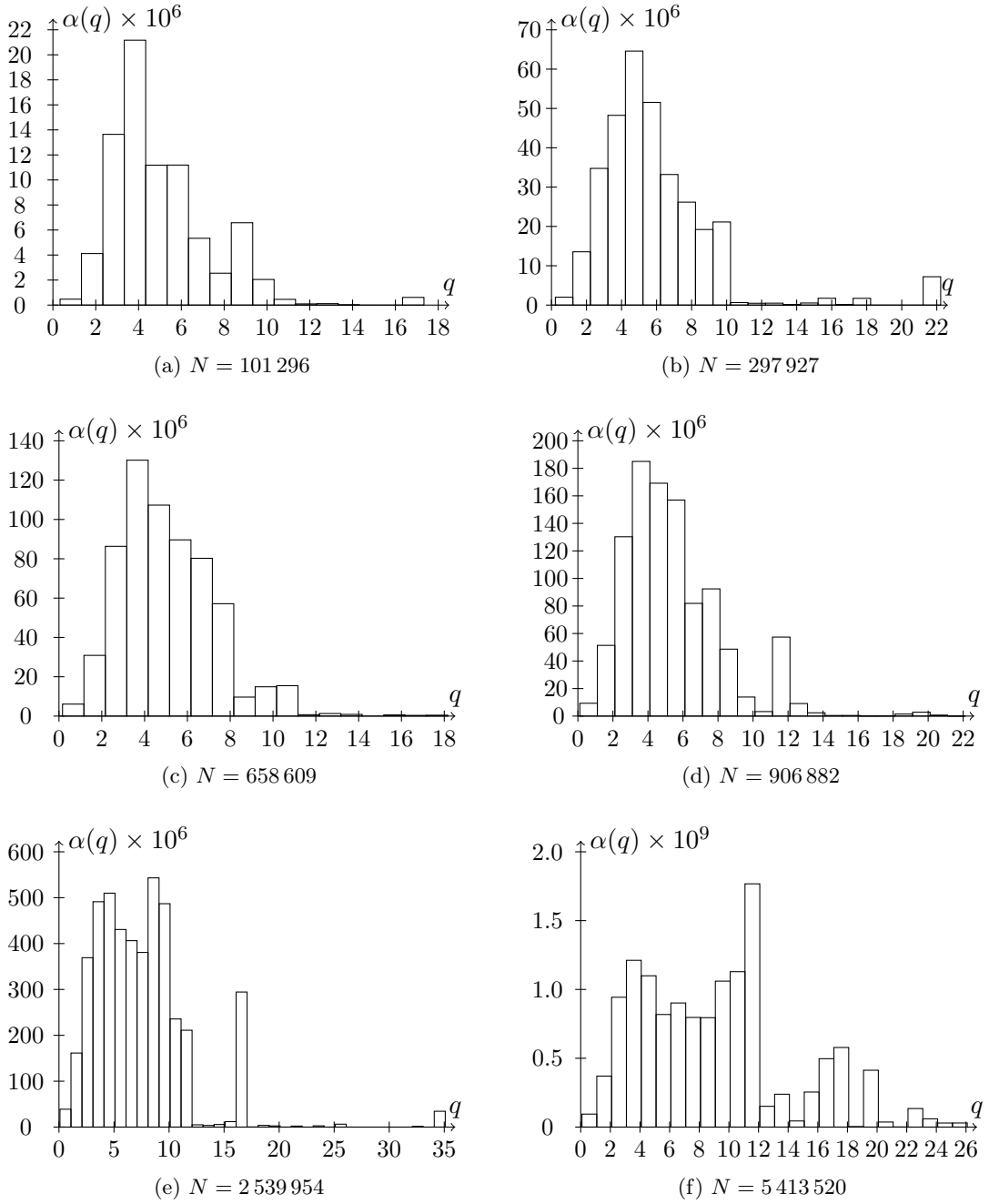


Abb. 5.16: Histogramme für die Abstandsberechnung in G_D^N

5.6.2.2 Kantengewichte nach virtueller Tiefe

Die Größe des Durchmessers eines Clusters hängt von der Ebene des Clusterbaums ab, in welcher dieser durch Teilung erzeugt wurde. Der Clustergraph G_D enthält nur Knoten

der ℓ-ten Ebene des Clusterbaums. Die Knoten werden aber in verschiedenen Ebenen generiert. Während die Durchmesser von Clustern, die aus der Teilung in der $(\ell-1)$-sten Ebene hervorgehen, ähnlich groß sind, sind die Durchmesser virtueller Knoten größer als die der nicht virtuellen Knoten.

Führt der Weg über einen virtuellen Knoten v, ist die Unsicherheit bzgl. $|p_{|v}|$ besonders groß. Deshalb wird ein kürzester Weg, der virtuelle Knoten enthält, durch die Wahl von entsprechenden Kantengewichten verlängert. Dazu wird die Gewichtsfunktion

$$w(t_1, t_2) = \max\{\mathrm{lev}\,(t_1) - \mathrm{lev}^{virt}(t_1)\,, \mathrm{lev}\,(t_2) - \mathrm{lev}^{virt}(t_2)\,, 1\}$$

für Kanten eingeführt, wobei $\mathrm{lev}\,(t)$ die aktuelle (ℓ-te) Ebene des Clusterbaums ist und $\mathrm{lev}^{virt}(t)$ die Ebene des Clusterbaums ist, in der der Cluster t durch Teilung erzeugt wurde. Der mit diesen Kantengewichten versehene Graph wird mit G_D^V bezeichnet. In G_D^V wird zur Berechnung des kürzesten Weges zwischen zwei Knoten anstatt der Breitensuche der Algorithmus von Dijkstra genutzt.

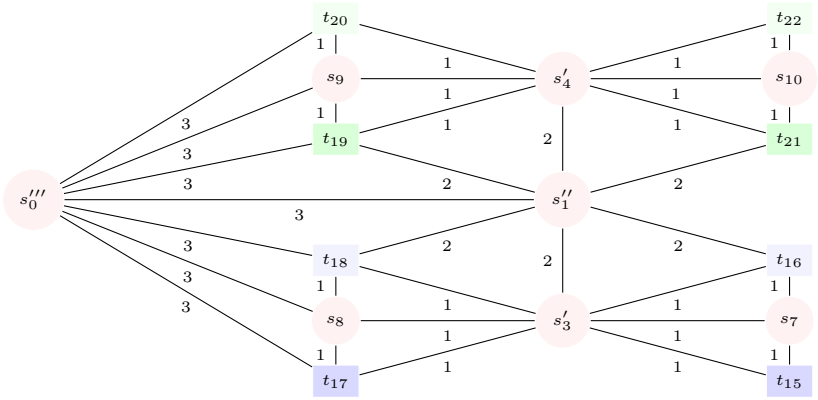


Abb. 5.17: G_D Kantengewichte nach virtueller Tiefe

Im Folgenden wird die Methode anhand der FE-Matrizen des Beispiels 5.14 (Tab. 5.1) getestet. Zur Konstruktion des Block-Clusterbaums wird das oben beschriebene Verfahren zur approximativen Abstandsbestimmung eingesetzt. Zusätzlich zum approximativen Abstand wird der exakte Abstand für jeden Block bestimmt und der Quotient (5.25) gebildet. In der Tab. 5.5 sind die prozentualen Anzahlen in den Intervallen der Intervallpartition (5.26) für $\mu = 2$ dargestellt. In dem Maße wie die Werte in der zweiten Spalte zunehmen, nehmen die Werte in der dritten Spalte ab. In der Asymptotik wird die Partition eher zu fein. Wie aus der letzten Spalte ersichtlich wird, nimmt der Anteil der Blöcke, für die das Verhältnis zwischen Abstandsnäherung und exaktem Abstand größer als 2.0 ist, tendenziell für größer werdende Matrizen ab.

Da einige Quotienten q stärker von Eins abweichen, muss zur genaueren Abschätzung des Aufwands der Berechnung der Matrixpartition die Blockgröße berücksichtigt werden. Hierzu wird die Funktion (5.27) genutzt. Der Verlauf der Funktion $\alpha(q)$ ist für die Testbeispiele aus Tab. 5.1 in den Abb. 5.18a bis 5.18f dargestellt. In jeder der Abbildungen ist ein Abfall der Kosten bei steigendem q zu beobachten. Daher gibt es für großes q

Matrix-größe	$\lvert\{q \in [q_{\min}, 0.5)\}\rvert$ in %	$\lvert\{q \in [0.5, 2.0)\}\rvert$ in %	$\lvert\{q \in [2.0, q_{\max}]\}\rvert$ in %
101 296	28.02	67.61	4.37
297 927	29.79	67.66	2.55
658 609	31.86	65.95	2.20
906 882	34.52	62.89	2.60
2 539 954	43.67	55.32	1.01
5 413 520	40.00	58.88	1.12

Tabelle 5.5: Prozentualer Anteil der Quotienten in den Intervallen $[q_{\min}, 0.5)$, $[0.5, 2.0)$ sowie $[2.0, q_{\max}]$ falls der Abstand in G_D^V bestimmt wird.

1. nur wenige Blöcke und
2. sind diese Blöcke relativ klein.

Folglich muss der Rang nur für wenige und relativ kleine Blöcke um den Faktor q^d erhöht werden. Zudem lassen die Abbildungen vermuten, dass q durch eine Schranke von moderater Größe beschränkt ist. Des Weiteren ähnelt der Funktionsverlauf der größeren Beispiele einem exponentiellen Abfall.

Im Abschnitt 6.2.2 wird die gesamte Setup-Phase zur Erstellung eines Vorkonditionierers auf Basis der hierarchischen LU-Zerlegung zeitlich aufgeschlüsselt. Es zeigt sich, dass die Zeit zur Konstruktion des Block-Clusterbaums nur einen kleinen Teil der gesamten Setup-Zeit ausmacht. Zudem kann der Vorkonditionierer auf Basis der algebraischen Partition die Anzahl der Iterationen des CG-Verfahrens deutlich reduzieren.

5.6.2.3 Verfeinerungsmethode

Die folgende Methode basiert nicht, wie die in den Abschnitten 5.6.2.1 und 5.6.2.2 beschriebenen Methoden, auf der Wahl von Gewichten für die Kanten in G_D, sondern auf der rekursiven Verfeinerung von in G_D berechneten Wegen.

Seien t_1, t_2 zwei Cluster der ℓ-ten Ebene des Clusterbaums zwischen denen der Abstand gesucht ist. Um das erste Problem der Bemerkung 5.21 abzumindern, werden m verschiedene Wege $p_1(t_1, t_2), \ldots, p_m(t_1, t_2)$ mit der Breitensuche in G_D berechnet. Das Problem zwei der Bemerkung 5.21 soll durch eine Verfeinerung der Wege abgemildert werden. Im folgenden betrachten wir o.B.d.A. den Fall $m = 1$.

Der Algorithmus zur Breitensuche wird so modifziert, dass auch die Knoten des Weges gespeichert werden. Zur Verbesserung der Abstandsberechnung wird ausgenutzt, dass die Knoten eines Weges $p(t_1, t_2) = (u_0, \ldots, u_i)$ in G_D im Allgemeinen wieder aus Teilgraphen des Matrixgraphen bzw. Clustern bestehen. Zu $p(t_1, t_2)$ wird ein Graph $G_D^{p(t_1,t_2),\ell_{\text{refine}}}$ konstruiert, dessen Knotenmenge aus den Knoten der Ebene ℓ_{refine} der Teilbäume mit den Wurzeln $u_0, \ldots, u_i$ besteht. Die Kantenmenge wird durch die Nachbarschaften induziert.

Ist die ℓ-te Ebene nahe den Blättern oder die Blattebene selbst, d.h. $\ell + \ell_{\text{refine}} \geq L(T_I)$, so ist die eben beschriebene Methode der Verfeinerung nicht anwendbar. Allerdings sind Cluster $t_1, t_2 \in \mathcal{L}(T_I)$ relativ klein und der entsprechende Ausschnitt des Matrixgraphen kann direkt betrachtet werden.

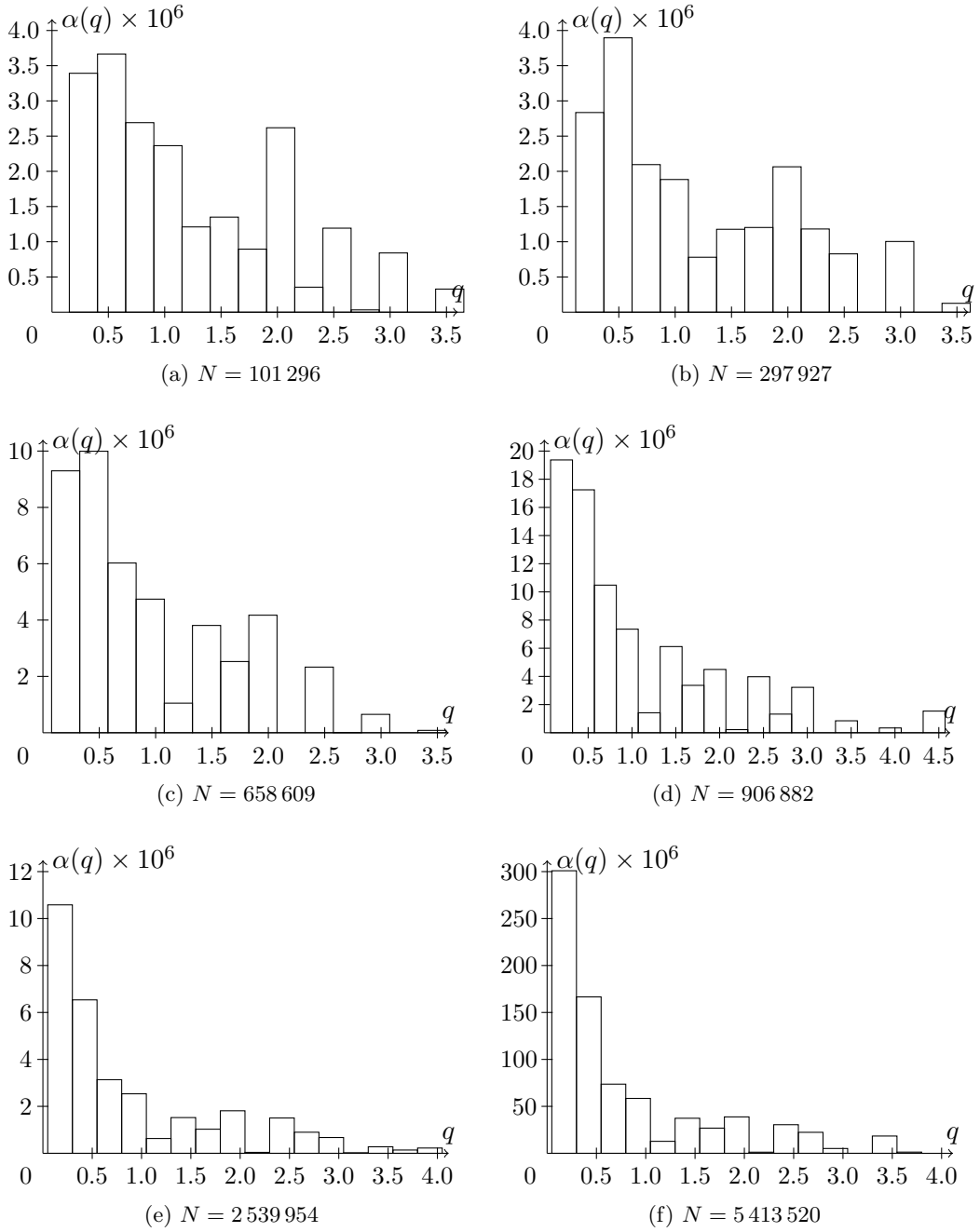


Abb. 5.18: Histogramme für die Abstandsberechnung in G_D^V - verschiedene Diskretisierungen

Wir berechnen einen kürzesten Weg innerhalb des Graphen $G_D^{p(t_1,t_2),\ell_{\text{refine}}}$, nachdem entsprechende Start- und Zielknoten $t_{src} \in t_1$ und $t_{target} \in t_2$ bestimmt wurden, siehe

Alg. 20. Der Algorithmus kann für den Fall $m > 1$ durch eine entsprechende Schlei-

Algorithmus 20 approxDist (t_1, t_2)

1: Konstruiere G_D (t_1, t_2)
2: berechne $p(t_1, t_2) = (u_0, , \ldots, u_i)$
3: **if** $|p(t_1, t_2)| < \eta \cdot c$ **then**
4: $\quad \Delta\ell = L(T_I) - \mathrm{lev}\,(t_1)$
5: $\quad$ **if** $\Delta\ell < \ell_{\text{refine}}$ **then**
6: $\quad\quad V(G_D^{p(t_1,t_2),\Delta\ell}) = \{v \in V(G_A) : v \in V(u_j), j = 0, \ldots, i\}$
7: $\quad\quad E(G_D^{p(t_1,t_2),\Delta\ell}) = \{(v, v') \in E(G_A) : v, v' \in V(G_D^{p(t_1,t_2),\Delta\ell})\}$
8: $\quad$ **else**
9: $\quad\quad V(G_D^{p(t_1,t_2),\ell_{\text{refine}}}) = \bigcup_{j=0}^{i}\{v \in T_{u_j}^{(\ell_{\text{refine}})}\}$
10: $\quad\quad E(G_D^{p(t_1,t_2),\ell_{\text{refine}}}) = \{(v', v'') : v', v'' \in V(G_D^{p(t_1,t_2),\ell_{\text{refine}}}), v', v''$ benachbart$\}$
11: $\quad$ **return** $|p_{G_D^{p(t_1,t_2),\ell_{\text{refine}}}}(t_{src}, t_{target})|$
12: **else**
13: $\quad$ **return** $|p(t_1, t_2)|$

fe ergänzt werden. Die neue Abstandsapproximation ist dann das Minimum aus den m Rechnungen. Zudem kann der Algorithmus mit geringer Modifikation rekursiv angewendet werden. Dazu wird in Zeile 11 nicht der Abstand zurück gegeben, sondern der Algorithmus wird erneut aufgerufen.

Beispiel 5.24. Angenommen die Approximation des in Beispiel 5.19 berechneten Abstands zwischen t_{15} und s_0''' soll verbessert werden. Zur Verfeinerung des Weges $p(t_{15}, s_0''')$ für $\ell_{\text{refine}} = 2$ werden die in Abb. 5.19 skizzierten Teil-Clusterbäume verwendet. Es ent-

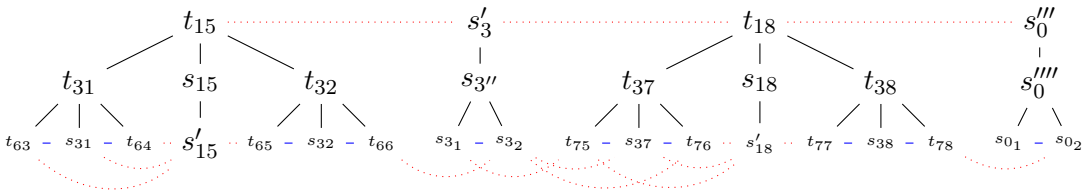


Abb. 5.19: Teil-Clusterbäume

steht der verfeinerte Graph in Abb. 5.20. Eine neue Approximation für den Abstand zwischen t_{15} und s_0''' wird beispielsweise durch den Abstand zwischen t_{64} und s_{0_1} gewonnen.

Wie im vorhergehenden Abschnitt liegt der Tab. 5.6 die Intervallpartition (5.26) des Intervalls $[q_{\min}, q_{\max}]$ für $\mu = 2$ zu Grunde. Die Werte der zweiten Spalte sind im Vergleich zur dritten und vierten Spalte klein. Die Partition ist folglich nur in sehr wenigen Blöcken zu fein. Die Werte der vierten Spalte sind deutlich größer als die Werte der vierten Spalte der Tab. 5.5. Die Partition wird daher teilweise zu grob.

Zur detaillierteren Untersuchung der Folgen der Überabschätzung des Abstands wird

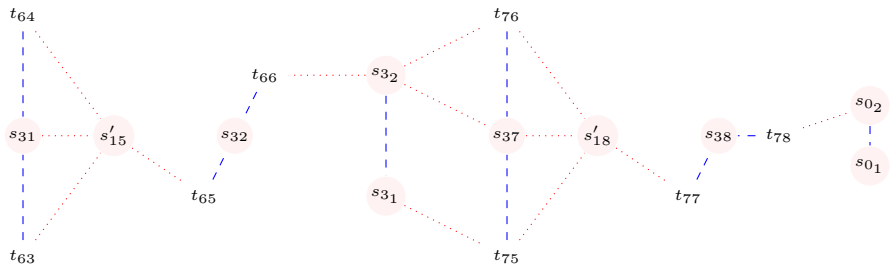


Abb. 5.20: Verfeinerter Graph

Matrix-größe	$\lvert\{q \in [q_{\min}, 0.5)\}\rvert$ in %	$\lvert\{q \in [0.5, 2.0)\}\rvert$ in %	$\lvert\{q \in [2.0, q_{\max}]\}\rvert$ in %
101 296	0.70	77.87	21.44
297 927	0.89	66.24	32.86
658 609	0.61	68.88	30.51
906 882	0.65	69.53	29.82
2 539 954	0.60	62.60	36.80
5 413 520	0.49	65.72	33.79

Tabelle 5.6: Verfeinerungsmethode, Anteil der Quotienten in den Intervallen $[q_{\min}, 0.5)$, $[0, 5, 2.0)$ sowie $[2.0, q_{\max}]$

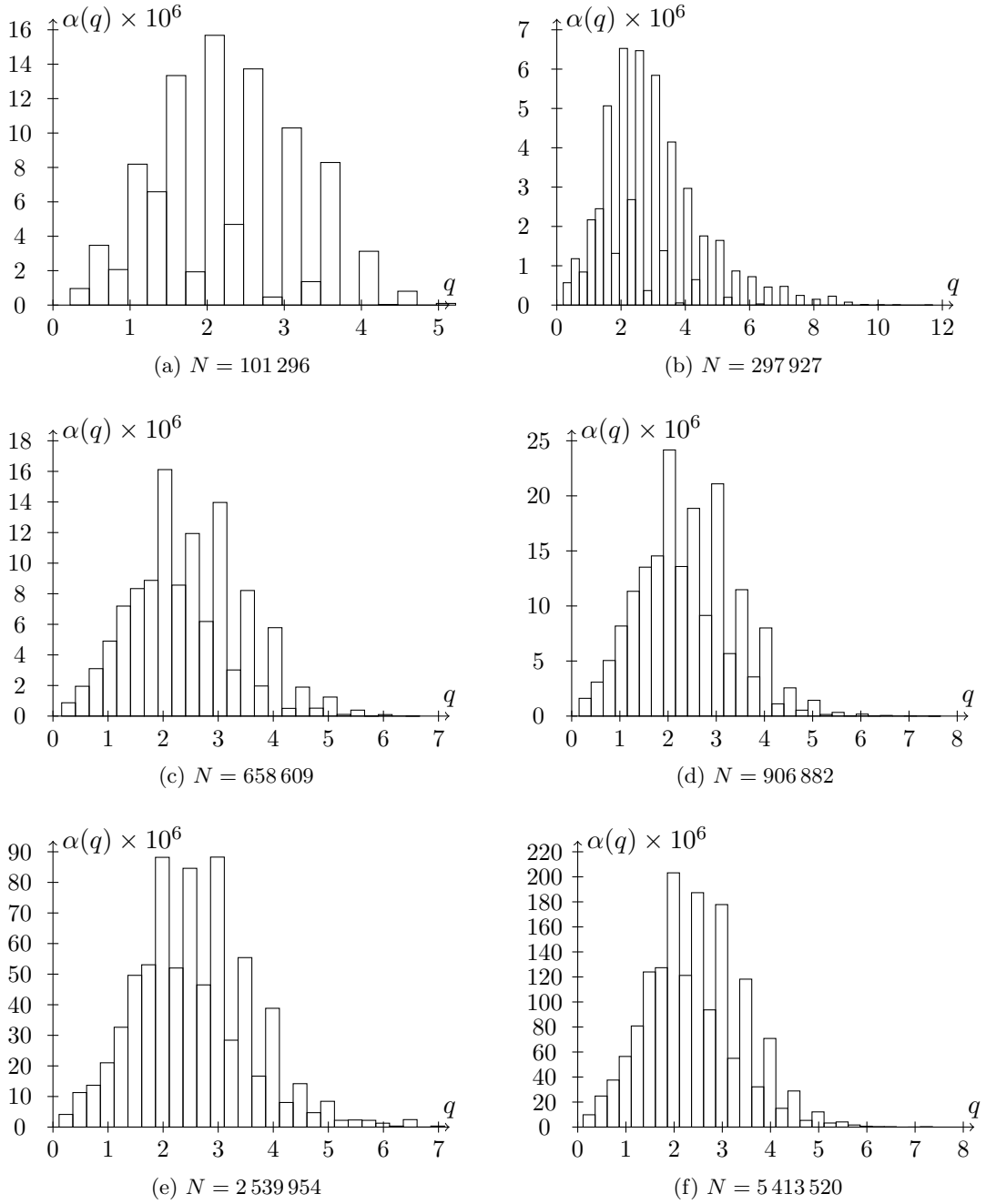
die Funktion (5.27) eingesetzt. Der Verlauf der Funktion $\alpha(q)$ ist für die Testbeispiele aus Tab. 5.1 in den Abb. 5.21a bis 5.21f dargestellt. Im Gegensatz zur Methode des Abschnitts 5.6.2.1 spiegelt sich die grobe Partition in den Abbildungen nicht wider. Die rechte Grenze des Intervalls, $q_{\max}$, ist deutlich kleiner als in den Abb. 5.16. Die Funktion $\alpha(q)$ erreicht in einem kleinen Intervall um 2 ihre maximalen Werte. Danach nimmt $\alpha(q)$ eher ab.

5.6.3 Abgeschwächte Zulässigkeit

Das Ziel der in [42, Abschnitt 2.5] eingeführten *schwachen Zulässigkeitsbedingung* besteht in einer Vergröberung der Partition, so dass die in den asymptotischen Komplexitätsabschätzungen enthaltenen Konstanten verkleinert werden. Nach dieser Zulässigkeitsbedingung ist ein Block $t_1 \times t_2$ zulässig, falls $t_1 \neq t_2$ ist oder der Block $t_1 \times t_2$ klein ist. Diese Bedingung kann in $\mathcal{O}(1)$ überprüft werden. Der zitierte Artikel beschränkt sich auf die Behandlung des ein-dimensionalen Falls. Zudem wird der Block-Clusterbaum auf geometrischer Basis erstellt.

Numerische Experimente für den mehrdimensionalen Fall und algebraischer Matrixpartitionierung zeigen, dass diese einfache Bedingung nicht ohne weiteres übertragen werden kann. Zwar kann die Partition im Vergleich zu den anderen hier vorgestellten Verfahren sehr schnell berechnet werden, aber der Vorkonditionierer auf Basis dieser Matrixpartition ist nicht so effizient wie die Vorkonditionierer auf Basis der bereits vorgestellten Matrixpartitionen.

Die im Artikel [42] vorgeschlagene Bedingung schränkt die Matrixpartition zu sehr ein.



(a) $N = 101\,296$ (b) $N = 297\,927$

(c) $N = 658\,609$ (d) $N = 906\,882$

(e) $N = 2\,539\,954$ (f) $N = 5\,413\,520$

Abb. 5.21: Histogramme für approximative Abstandsberechnung mittels Refinement-Strategie für verschiedene Diskretisierungen

Ein Mittelweg zwischen der schwachen Zulässigkeitsbedingung und der Bedingung (5.4) ist die *abgeschwächte Zulässigkeit*. Nach dieser ist ein Block $t_1 \times t_2$, $t_1 \neq t_2$, zulässig, falls

die Indexmengen t_1 und t_2 nicht benachbart sind. Es müssen keine Abstände berechnet werden. In Kapitel 6.2.4 werden wir sehen, dass mit dieser Bedingung eine Partition generiert wird, mit welcher in den meisten Fällen die Berechnung eines effizienten Vorkonditionierers möglich ist. In der Tab. 6.7 ist ein Beispiel angegeben, bei dem nicht auf eine approximative Abstandsberechnung verzichtet werden kann.

5.7 Berechnung der $\mathcal{H}$-LU-Zerlegung

Die aus der Nested-Dissection Umordnung resultierende Blockstruktur von A bzw. der Faktoren L und U erlaubt eine effiziente Parallelisierung der Zerlegung. In einem ersten Schritt können die Blöcke $A_{t_1t_1}$ und $A_{t_2t_2}$ unabhängig voneinander faktorisiert werden:

$$A_{t_1t_1} = L_{t_1t_1}U_{t_1t_1}, \quad A_{t_2t_2} = L_{t_2t_2}U_{t_2t_2}.$$

Sind die Blöcke $A_{t_it_i}$, $i = 1, 2$, noch ausreichend groß, dann wird die LU-Zerlegung dieser Blöcke rekursiv (und parallel) berechnet.

Im zweiten Schritt werden die Blöcke U_{t_1s}, U_{t_2s} durch $\mathcal{H}$-Vorwärtssubstitution und die Blöcke L_{st_1}, L_{st_2} durch $\mathcal{H}$-Rückwärtssubstitution bestimmt. Da die Berechnungen unabhängig von einander sind, können die Einträge der Blöcke parallel ermittelt werden. Im letzten Schritt müssen die Blöcke L_{ss} und U_{ss} berechnet werden:

$$L_{ss}U_{ss} = A_{ss} - L_{t_1s}U_{st_1} - L_{t_2s}U_{st_2}.$$

Da dieser Schritt nicht parallel ausgeführt wird, ist es besonders wichtig, einen kleinen Separator zu finden, um den seriellen Anteil des Algorithmus zu reduzieren.

6 Numerische Experimente

Zur algebraischen Partitionierung werden weniger Informationen benötigt als zur geometrischen Partitionierung. Deshalb kann man erwarten, dass der mit geometrischen Informationen konstruierte Vorkonditionierer effizienter ist. Andererseits werden bei der algebraischen Partitionierung die Matrixeigenschaften besser ausgenutzt. Ein Vergleich von Vorkonditionierern basierend auf geometrischer und algebraischer Partitionierung wird im Abschnitt 6.1 vorgenommen.

Die numerischen Experimente im Abschnitt 6.2 sollen Aussagen zum asymptotischen Laufzeitverhalten der im vorigen Kapitel entwickelten Algorithmen liefern. Dazu werden die linearen Probleme des Beispiels 5.14 (Tab. 5.1) gelöst.

Die Robustheit der hierarchischen Matrizen wird im Abschnitt 6.3 anhand der Lösung von Problemen mit schlechter Kondition belegt.

Die auf der Nested-Dissection-Clusterung basierende Partition erlaubt eine Parallelisierung der Teilaufgaben. Ergebnisse für mit der parallelisierten $\mathcal{H}$-Cholesky-Zerlegung gerechnete Beispiele enthält Abschnitt 6.5.

Im Rahmen der Lösung nichtlinearer Probleme mit Hilfe des Newton-Verfahrens ist eine Folge von linearen Gleichungssystemen zu lösen. Ist die (hierarchische) LU-Zerlegung bzw. die (hierarchische) Cholesky-Zerlegung berechnet, kann diese zur Lösung mehrerer rechter Seiten genutzt werden. Diese Eigenschaft wird in einem vereinfachten Newton-Verfahren ausgenutzt. Die Ergebnisse befinden sich im Abschnitt 6.6.

Falls die Testumgebung nicht explizit beschrieben wird, wird in den folgenden Abschnitten für jede FE-Matrix ein Vorkonditionierer berechnet und das Gleichungssystem mit Hilfe des vorkonditionierten CG-Verfahrens gelöst. Das CG-Verfahren wird beendet, sobald die Spektralnorm des Residuums die Bedingung $\|\boldsymbol{r}\|_2 \leq 10^{-5} \|\boldsymbol{b}\|_2$ erfüllt.

Für die Rechnungen wurde ein Intel Xeon mit acht Prozessorkernen (3.2 GHz, 6 MB Cache pro Kern) und 64 GB Hauptspeicher eingesetzt.

6.1 Vergleich von geometrischer und algebraischer Matrixpartition

Im Folgenden wird der auf einer rein algebraisch konstruierten Matrixpartition basierende Vorkonditionierer *algebraischer Vorkonditionierer* genannt. Liegt dem Vorkonditionierer ein auf Basis von geometrischen Informationen konstruierter Block-Clusterbaum zu Grunde, so wird dieser im Weiteren als *geometrischer Vorkonditionierer* bezeichnet.

Zum Vergleich des geometrischen und des algebraischen Vorkonditionierers werden die folgenden Kriterien herangezogen:

1. die Zeit zur Konstruktion des Vorkonditionierers,

2. der Speicherverbrauch des Vorkonditionierers sowie

3. die Wirkung des Vorkonditionierers in einem iterativen Verfahren.

Als Testbeispiele dienen die Matrizen aus Tab. 5.1. Zur geometrischen Partitionierung

	$\mathcal{H}$-Cholesky-Zerlegung und PCG-Verfahren								
	P_{geo} basiert				P_{alg} basiert				
	Setup		PCG		Setup		PCG		CG
Matrix-	Speicher	Zeit	#it	Zeit	Speicher	Zeit	#it	Zeit	#it
größe	[MB]	[s]		[s]	[MB]	[s]		[s]	
F	128	68.44	8	2.16	49	5.48	10	0.97	84
M1	364	371.87	11	11.59	183	20.37	12	4.02	131
VF	1 358	1 176.04	13	36.02	443	47.91	14	11.21	164
C2	1 993	1 797.05	16	64.70	551	76.96	18	20.66	241
M2	6 704	6 945.23	20	262.04	1 841	253.78	23	85.48	313
VF1	16 897	18 943.10	26	833.71	4 148	611.64	30	250.43	415

Tabelle 6.1: Vergleich von geometrischer mit algebraischer Partitionierung

wurde das PCA-Verfahren eingesetzt. Die Werte in Tab. 6.1 dokumentieren sowohl den geringeren Speicherverbrauch als auch eine geringere Konstruktionszeit des algebraischen Vorkonditionierers. Trotz der leicht höheren Anzahl an Iterationen des algebraisch vorkonditionierten CG-Verfahrens (achte vs. vierte Spalte) wird zur Lösung weniger Zeit (Spalte neun) benötigt als mit dem geometrisch vorkonditionierten CG-Verfahren (Spalte fünf). Auf Grund der größeren Speicherbelegung des geometrischen Vorkonditionierers ist dessen Anwendung aufwändiger.

Zur Beurteilung der Reduktion der Iterationszahl des CG-Verfahrens sind in der letzten Spalte der Tabelle die Anzahl der Iterationen des CG-Verfahrens ohne Vorkonditionierer angegeben.

Ein weiterer Vorteil für den algebraischen Vorkonditionierer ist die gute Parallelisierbarkeit der $\mathcal{H}$-LU-Zerlegung, siehe Abschnitt 6.5.

6.2 Vergleich der algebraischen Matrixpartitionen

In diesem Abschnitt werden Ergebnisse zur Asymptotik des Setups der auf den verschiedenen algebraischen Matrixpartitionen beruhenden Vorkonditionierer präsentiert sowie Tests zur Wirkung des jeweiligen Vorkonditionierers durchgeführt.

Die rekursive Clusterung wird gestoppt, sobald der Cluster die Größe $n_{\min} = 50$ unterschreitet. Die blockweise Genauigkeit wird auf $\delta = 0.1$ gesetzt.

6.2.1 Mengenschnitt

Die Mengenschnitt-Partition wird nach dem in Abschnitt 5.6.1 beschriebenen Verfahren konstruiert. Den auf Basis dieser Partition berechneten Vorkonditionierer nennen wir *Mengenschnitt-Vorkonditionierer*.

Im Vergleich zu den nachfolgenden Abstandsapproximationen wird der Wert für η in der Zulässigkeitsbedingung (5.4) relativ klein gewählt. Werden ähnlich große Werte für η wie in den Abschnitten 6.2.2 und 6.2.3 eingesetzt, so wird die Partition zu fein.

Nach [34] ist der Aufwand für einen Zulässigkeitstest $\mathcal{O}\left(d^d \max\{|t_1|, |t_2|\}\right)$. Dies zeigt sich auch in den Zeiten der fünften Spalte in der Tab. 6.2. Im Vergleich zu den anderen Bestandteilen des Setups ist der Aufbau von $T_{I\times I}$ die aufwändigste Operation. Der Vorkonditionierer reduziert die Anzahl der Iterationen in gleichem Maße wie die in den Abschnitten 6.2.2 und 6.2.3 untersuchten Vorkonditionierer. Die zum Aufbau dieser Vorkonditionierer verwendeten Block-Clusterbäume können jedoch in wesentlich kürzerer Zeit konstruiert werden.

Matrix	Cluster		Block-Cluster		Transformation		$\mathcal{H}$-Cholesky		PCG	
	#	t in s	#	t in s	t in s	MB	t in s	MB	#it	t in s
101 296	4 446	1.44	24 286	1.59	1.07	41	2.68	48	10	0.95
297 927	8 989	4.90	59 011	8.63	3.88	156	10.04	182	12	3.88
658 609	27 821	12.56	128 208	27.92	9.70	376	23.01	440	14	10.88
906 882	36 581	18.68	238 842	50.98	10.32	437	39.78	550	18	18.93
2 539 954	113 125	62.28	589 708	293.04	34.74	1 494	122.70	1 847	23	75.45
5 413 520	230 813	152.76	1 307 949	651.66	81.87	3 347	283.40	4 141	29	211.27

Tabelle 6.2: Auswertung Zulässigkeit mit Mengenschnitt, $\delta = 0.1$, $n_{\min} = 50$, $\eta = 0.5$

6.2.2 Kantengewichte nach virtueller Tiefe

Gegenstand der Untersuchung dieses Abschnitts ist die Gewichtswahl nach virtueller Tiefe im Clustergraphen. Die in der Tab. 6.3 angegebenen Daten schlüsseln die einzelnen Bestandteile der Konstruktion des Vorkonditionierers auf. Zudem enthält die Tabelle die Anzahl der Iterationen bis zur Konvergenz des CG-Verfahrens und dessen Laufzeit. In der Zulässigkeitsbedingung (5.4) wird $\eta = 16$ gesetzt. Die numerischen Experimente des Abschnitts 5.6.2.2 ergeben, dass die Abstandsapproximation eher zu einer feinen Partition führt. Tatsächlich ist die Anzahl der Blöcke in der Partition (vierte Spalte der Tab. 6.3) im Vergleich zur Anzahl der in Tab. 6.2 angegebenen Blöcke größer. Zur Konstruktion der Blockpartition wird jedoch wesentlich weniger Zeit benötigt als zur Mengenschnitt-Partition. Die weiteren Teile der Berechnung des Vorkonditionierers weisen ähnliche Laufzeiten wie die des Mengenschnitt-Vorkonditionierers auf. Auch die Wirkung des Vorkonditionierers ist zum Mengenschnitt-Vorkonditionierer analog. Vergleicht man die Gesamt-Rechenzeiten beider Methoden dann ergibt sich eine signifikante Einsparung für die auf der Wahl der Kantengewichte nach virtueller Tiefe basierenden Methode.

6.2.3 Verfeinerungsmethode

Für die numerischen Tests der Verfeinerungsmethode des Abschnitts 5.6.2.3 werden drei verschiedene Pfade ($m = 3$) berechnet. Diese werden zweimal rekursiv verfeinert. Seien t_1, t_2 zwei Cluster der ℓ-ten Ebene des Clusterbaums. Zur ersten Verfeinerung wird

Matrix	Cluster		Blockcluster		Transformation		$\mathcal{H}$-Cholesky		PCG	
	#	t in s	#	t in s	t in s	MB	t in s	MB	#it	t in s
101 296	4 446	1.77	24 773	0.04	1.07	41	2.66	48	10	0.96
297 927	8 989	5.64	62 318	0.20	3.88	156	10.07	182	12	3.92
658 609	27 821	14.17	134 918	0.45	9.70	376	23.18	439	14	11.04
906 882	36 581	21.67	254 866	1.82	10.35	437	40.49	549	18	19.23
2 539 954	113 125	69.11	658 607	18.16	34.77	1 495	126.38	1 843	23	77.17
5 413 520	230 813	167.21	1 437 235	41.00	82.17	3 350	294.60	4 135	29	215.81

Tabelle 6.3: Kantengewichte per virtueller Tiefe: $\delta = 0.1$, $n_{\min} = 50$, $\eta = 16$

ein Clustergraph aus Clustern der Ebene $\ell + 3$ aufgebaut, d.h. $\ell_{\text{refine}} = 3$. Zur zweiten Verfeinerung eines Weges in $G_D^{p(t_1,t_2),3}$ wird der entsprechende Clustergraph in der Ebene $\ell + 6$ berechnet.

In der Tab. 6.4 sind neben den Phasen der Konstruktion des Vorkonditionierers die Anzahl der Iterationen sowie die Laufzeit des CG-Verfahrens dargestellt. Die numerischen Tests des Abschnitts 5.6.2.3 zeigen, dass die berechnete Partition eher zu grob ist. Die zu grobe Partition spiegelt sich jedoch nicht in den Resultaten der Tab. 6.4 wider. In den Testbeispielen der Tabelle werden zwar deutlich weniger Blöcke berechnet (Spalte vier) als mit dem vorhergehenden Verfahren (vgl. Tab. 6.3), aber sowohl die Laufzeit zur Transformation der Matrixeinträge in die entsprechenden Blöcke als auch die Zeit zur hierarchischen Cholesky-Zerlegung sind denen der vorhergehenden Methoden ähnlich. Da die mittels der Verfeinerungsmethode berechnete Matrixpartition von allen bisher betrachteten Methoden die geringste Laufzeit benötigt, sollte diese in praktischen Anwendungen verwendet werden.

Matrix	Cluster		Blockcluster		Transformation		$\mathcal{H}$-Cholesky		PCG	
	#	t in s	#	t in s	t in s	MB	t in s	MB	#it	t in s
101 296	4 446	1.70	24 315	0.11	1.07	41	2.69	48	10	0.97
297 927	8 989	5.76	59 177	0.30	3.89	156	10.10	182	12	3.94
658 609	27 821	14.22	128 437	0.83	9.74	376	23.09	440	14	11.10
906 882	36 581	21.35	239 361	1.50	10.39	437	39.98	549	18	19.28
2 539 954	113 125	69.40	591 836	4.58	34.91	1 494	122.73	1 842	23	76.91
5 413 520	230 813	168.25	1 311 478	10.12	82.34	3 347	284.39	4 128	29	215.26

Tabelle 6.4: Verfeinerungsmethode: $\delta = 0.1$, $n_{\min} = 50$, $\eta = 16$, $m = 3$, $\ell_{\text{refine}} = 3$, zweifache Rekursion

Die deutliche Reduktion der Anzahl der Iterationen des CG-Verfahrens im Vergleich zum nicht vorkonditionierten CG-Verfahren (siehe letzte Spalte der Tab. 6.1) belegt die Effizienz des Vorkonditionierers.

6.2.4 Schwache und abgeschwächte Zulässigkeit

Für die folgenden Experimente wird die Partition anhand der in [42, Abschnitt 2.5] eingeführten schwachen Zulässigkeitsbedingung konstruiert. Wie die Resultate in der Tab. 6.5 zeigen, kann der auf dieser Partition berechnete Vorkonditionierer zwar in sehr kurzer

Zeit und mit sehr geringem Speicherbedarf berechnet werden, reduziert aber die Anzahl der Iterationen im Vergleich zu den bisherigen Methoden nur ungenügend. Schwächt man

Matrix	Cluster		Blockcluster		Transformation		$\mathcal{H}$-Cholesky		PCG	
	#	t in s	#	t in s	t in s	MB	t in s	MB	#it	t in s
101 296	4 446	1.53	5 412	0.00	0.03	11	0.09	11	49	1.20
297 927	8 989	5.25	11 184	0.00	0.09	45	0.39	45	61	6.09
658 609	27 821	13.65	23 183	0.01	0.20	107	0.94	107	71	17.17
906 882	36 581	20.12	44 254	0.01	0.28	106	0.88	106	96	27.08
2 539 954	113 125	67.68	92 988	0.03	0.79	386	3.37	386	125	114.53
5 413 520	230 813	166.88	191 541	0.07	1.73	865	7.66	865	157	317.30

Tabelle 6.5: schwache Zulässigkeit: $\delta = 0.1$, $n_{\min} = 50$

die Zulässigkeitsbedingung nicht ganz so rigoros ab, sondern berechnet die Partition unter Berücksichtigung der Nachbarschaften (siehe Abschnitt 5.6.3), so bleibt der Vorkonditionierer meist so effizient wie bei den vorangegangenen Methoden, siehe Tab. 6.6. Die

Matrix	Cluster		Blockcluster		Transformation		$\mathcal{H}$-Cholesky		PCG	
	#	t in s	#	t in s	t in s	MB	t in s	MB	#it	t in s
101 296	4 446	1.44	24 286	0.00	1.07	41	2.67	48	10	0.95
297 927	8 989	4.94	59 011	0.01	3.88	156	10.05	182	12	3.88
658 609	27 821	12.65	128 208	0.03	9.69	376	23.00	440	14	10.89
906 882	36 581	18.76	238 842	0.05	10.32	437	39.73	550	18	18.91
2 539 954	113 125	62.43	589 708	0.17	34.68	1 494	122.05	1 847	23	75.44
5 413 520	230 813	153.42	1 307 949	0.37	81.89	3 347	283.36	4 141	29	211.36

Tabelle 6.6: abgeschwächte Zulässigkeit: $\delta = 0.1$, $n_{\min} = 50$

vierte Spalte der Tabelle zeigt, dass die Konstruktion des Block-Clusterbaums selbst für das Beispiel mit feinster Diskretisierung weniger als eine Sekunde benötigt. Die Block-Clusterung hat folglich nur noch einen sehr geringen Anteil an der Konstruktionszeit des Vorkonditionierers.

In dem folgenden numerischen Experiment wird die $\mathcal{H}$-Cholesky-Zerlegung der Matrix boneS01 aus der Matrixgruppe Oberwolfach (id 1454) der Matrixsammlung der Universität Florida (University of Florida Sparse Matrix Collection[1]) verwendet. In der Tab. 6.7 wird die Partition basierend auf der abgeschwächten Zulässigkeit berechnet. Die Resultate der Tab. 6.8 wurden durch Anwendung der im Abschnitt 5.6.2.2 vorgestellten Methode zur Partitionierung gewonnen. Die nach der Methode Kantengewichten per virtueller Tiefe berechnete Partition ist für dieses Beispiel robuster als die mit der abgeschwächten Zulässigkeit berechnete Partition.

Bemerkung 6.1. Bis auf die Partition, welche auf der schwachen Zulässigkeitsbedingung basiert, eignen sich alle untersuchten Partitionen zur Konstruktion eines effizienten Vorkonditionierers. Im Abschnitt 5.1 ist

$$\operatorname{diam} \partial_{t_1} t_2 \leq \eta \operatorname{dist}(t_1, t_2) \quad \text{für ein} \quad \eta > 0$$

[1] `http://www.cise.ufl.edu/research/sparse/matrices`

Matrix	δ	Cluster		Blockcluster		Transformation		$\mathcal{H}$-Cholesky		PCG	
		#	t in s	#	t in s	t in s	MB	t in s	MB	#it	t in s
127 224	$5 \cdot 10^{-1}$	4 495	5.42	26 645	0.01	2.48	75	2.88	63	666	89.81
127 224	$1 \cdot 10^{-1}$	4 495	5.37	26 645	0.01	2.48	75	–	–	–	–
127 224	$5 \cdot 10^{-2}$	4 495	5.37	26 645	0.00	2.49	75	–	–	–	–
127 224	$1 \cdot 10^{-2}$	4 495	5.42	26 645	0.01	2.49	75	18.15	134	40	8.93
127 224	$5 \cdot 10^{-3}$	4 495	5.42	26 645	0.01	2.49	75	–	–	–	–
127 224	$1 \cdot 10^{-3}$	4 495	5.38	26 645	0.00	2.51	75	–	–	–	–
127 224	$5 \cdot 10^{-4}$	4 495	5.37	26 645	0.01	2.51	75	–	–	–	–
127 224	$1 \cdot 10^{-4}$	4 495	5.37	26 645	0.00	2.54	75	49.89	196	5	1.49

Tabelle 6.7: abgeschwächte Zulässigkeit: $n_{\min} = 50$

Matrix	δ	Cluster		Blockcluster		Transformation		$\mathcal{H}$-Cholesky		PCG	
		#	t in s	#	t in s	t in s	MB	t in s	MB	#it	t in s
127 224	$5 \cdot 10^{-1}$	4 495	5.38	52 059	39.05	2.51	76	3.56	64	665	97.66
127 224	$1 \cdot 10^{-1}$	4 495	5.33	52 059	39.00	2.52	76	–	–	–	–
127 224	$5 \cdot 10^{-2}$	4 495	5.33	52 059	39.17	2.52	76	–	–	–	–
127 224	$1 \cdot 10^{-2}$	4 495	5.33	52 059	39.04	2.53	76	19.85	129	35	8.19
127 224	$5 \cdot 10^{-3}$	4 495	5.33	52 059	39.00	2.52	76	24.74	139	22	5.44
127 224	$1 \cdot 10^{-3}$	4 495	5.33	52 059	39.12	2.54	76	36.50	159	11	3.01
127 224	$5 \cdot 10^{-4}$	4 495	5.33	52 059	39.02	2.54	76	41.65	167	8	2.28
127 224	$1 \cdot 10^{-4}$	4 495	5.33	52 059	39.12	2.57	76	52.02	183	5	1.53

Tabelle 6.8: Kantengewichte per virtueller Tiefe: $n_{\min} = 50$, $\eta = 4$

eine für den Beweis des Lemmas 5.2 ausreichende Voraussetzung. In den vorgestellten Algorithmen wird der approximierte Abstand in Relation zum minimalen Durchmesser der Indexmengen gesetzt. Wegen $\min\{\operatorname{diam} \partial_{t_1} t_2, \operatorname{diam} \partial_{t_1} t_2\} < \min\{\operatorname{diam} t_1, \operatorname{diam} t_2\}$ wird daher vermutlich generell eine zu feine Partition berechnet. Approximationen für den Durchmesser sind technisch jedoch wesentlich leichter zu bestimmen als die Werte $\operatorname{diam} \partial_{t_1} t_2$ bzw. $\operatorname{diam} \partial_{t_1} t_2$.

6.3 Vergleich von linearen Lösern: PARDISO, AMG-PCG und $\mathcal{H}$-PCG

Die Verfahren zur Lösung der bei der FE-Methode entstehenden schwach besetzten linearen Gleichungssysteme lassen sich grob in direkte und iterative Verfahren einteilen. PARDISO[2] [59, 60] ist nach [54] einer der aktuell schnellsten direkten Löser. Bei iterativen Verfahren wird zur Konvergenzbeschleunigung meist ein Vorkonditionierer eingesetzt. Als Vorkonditionierer wird in den numerischen Tests einerseits das Algebraische Mehrgitter-Verfahren des Softwarepakets hypre[3] benutzt, andererseits die im vorangegangenen Kapitel eingeführte hierarchische LU- bzw. Cholesky-Zerlegung.

Für asymptotische Aussagen werden die bereits im Kapitel 5 verwendeten Testmatrizen aus Tab. 5.1 des Beispiels 5.14 verwendet. Die Ergebnisse sind in Tab. 6.9 aufgeführt.

[2] aus Intel Math Kernel Library 10.2.2.025

[3] `https://computation.llnl.gov/casc/hypre/software.html`, Version 2.0.4b

In Übereinstimmung mit der Theorie wächst die Laufzeit direkter Verfahren für schwach

Matrix-größe	PARDISO			BoomerAMG				$\mathcal{H}$-Cholesky			
	Faktorisierung		Löser	Setup		Löser		Setup		Löser	
	Mem [MB]	Zeit [s]	Zeit [s]	Mem [MB]	Zeit [s]	#it	Zeit [s]	Mem [MB]	Zeit [s]	#it	Zeit [s]
101 296	257	9.64	0.30	136	1.52	6	1.08	48	5.19	10	0.95
297 927	955	48.36	1.87	447	5.09	6	3.51	182	18.88	12	3.88
658 609	2 937	251.66	2.93	1 136	13.96	7	10.25	440	45.37	14	10.89
906 882	4 420	438.08	3.45	1 831	21.34	8	18.45	550	68.86	18	18.91
2 539 954	17 128	3 374.42	13.06	5 644	69.17	8	55.79	1 847	219.33	23	75.44
5 413 520	49 556	18 574.60	35.93	10 925	142.50	9	122.79	4 141	519.04	29	211.36

Tabelle 6.9: Vergleich verschiedener Verfahren bzgl. Asymptotik

besetzte FE-Matrizen (dritten Spalte der Tabelle) quadratisch. Für M-Matrizen ist die Laufzeit des AMG-Verfahrens linear. Aus der Setup-Zeit des AMG-Verfahrens, welche sich in der sechsten Spalte der Tabelle befindet, lässt sich das lineare Verhalten ablesen. Die Zeit zur Konstruktion des hierarchischen Vorkonditionierers ist im Vergleich zur Setup-Zeit des AMG-Verfahrens höher. Im Gegensatz zum AMG-Verfahren muss aber der hierarchische Vorkonditionierer zur Lösung mehrerer rechter Seiten nicht jedesmal neu berechnet werden. Der Einsatz des hierarchischen Vorkonditionierers lohnt sich in den Testbeispielen bereits für wenige rechte Seiten (≤ 10).

Die Speicherbelegung des direkten Lösers wächst im Vergleich zu den beiden anderen Methoden sehr schnell an. AMG benötigt jedoch immer noch mehr als doppelt so viel Speicher wie die $\mathcal{H}$-Cholesky-Zerlegung.

Um die Leistungsfähigkeit der neuen Methode unter Beweis zu stellen, werden Probleme verschiedener numerischer Schwierigkeit aus der Matrixsammlung der Universität von Florida (University of Florida Sparse Matrix Collection) ausgewählt. Zur Lösung mit dem CG-Verfahren wird die Anzahl der maximalen Iterationen auf 1000 und die Genauigkeit auf 10^{-5} gesetzt. BoomerAMG wird mit den Standardeinstellungen benutzt.

Matrix	PARDISO			BoomerAMG				$\mathcal{H}$-Cholesky			
	Faktorisierung		Löser	Setup		Löser		Setup		Löser	
	Mem [MB]	Zeit [s]	Zeit [s]	Mem [MB]	Zeit [s]	#it	Zeit [s]	Mem [MB]	Zeit [s]	#it	Zeit [s]
Crystm03	43	2.40	0.16	–	–	–	–	15	1.87	5	0.12
Ct20stif	127	3.78	0.13	–	–	–	–	69	16.90	25	2.66
Crank_seq1	376	12.56	0.25	–	–	–	–	120	45.45	46	9.31
Crank_seq2	489	15.42	0.52	–	–	–	–	170	72.15	30	8.41

Tabelle 6.10: Vergleich verschiedener Verfahren bzgl. Robustheit

Während PARDISO alle Aufgaben der Tab. 6.10 bewältigt, kann das mit AMG (in der Standardkonfiguration) vorkonditionierte CG-Verfahren keines der ausgewählten Probleme lösen. Zur Konvergenz des hierarchisch vorkonditionierten CG-Verfahrens muss zum Teil die blockweise Genauigkeit erhöht werden.

Mit einer linear-logarithmischen Zeit- und Speicherkomplexität und der in Tab. 6.10

belegten Robustheit sind die hierarchischen Matrizen für ein großes Spektrum an Aufgaben geeignet.

6.4 $\mathcal{H}$-LU-Zerlegung

In den ersten Abschnitten dieses Kapitels wurden die Algorithmen an symmetrisch positiv definiten Problemen getestet. Dazu wurde die $\mathcal{H}$-Cholesky-Zerlegung eingesetzt. Bei dieser wird die Symmetrie und die positive Definitheit ausgenutzt. Bei unsymmetrischen Matrizen muss die Matrix statt in die Faktoren L und L^T in die Matrixfaktoren L und U zerlegt werden.

Zur Lösung von unsymmetrischen Problemen wird im Folgendem die $\mathcal{H}$-LU-Zerlegung als Vorkonditionierer des GMRes-Verfahrens [58] mit Restart (GMRes (m)) eingesetzt.

Zum Test der $\mathcal{H}$-LU-Zerlegung soll auf dem Berechnungsgebiet der Abb. 6.1 die Strom-

Abb. 6.1: Teil eines ABB-Geräts

erhaltung bei Gleichstrom simuliert werden.

Der Zusammenhang zwischen elektrischen und magnetischen Feldern sowie deren Erzeugung durch Ladungen und Ströme wird durch partielle Differentialgleichungen, die Maxwellschen Gleichungen, beschrieben.

An den drei Kontakten des Geräts der Abb. 6.1 liegen Dirichlet-Randbedingungen vor, am Rand des restlichen Gebietes Neumann-Randbedingungen. Zur Diskretisierung der Maxwellschen Gleichungen wurden FEM-Ansatz-Funktionen zweiter Ordnung verwendet.

Wird das GMRes(m)-Verfahren ohne Vorkonditionierer mit $m = 30$ auf die aus der Diskretisierung resultierenden Modellmatrizen angewandt, so konvergiert es nicht.

In der Tab. 6.11 sind die Ergebnisse für das $\mathcal{H}$-LU-vorkonditionierte GMRes(30)-Verfahren angegeben. Der Zeit- und Speicheraufwand zur Konstruktion des Vorkonditionierers ist verglichen mit den in den Abschnitten 6.2.2 bis 6.2.4 angegebenen Werten für

Matrix-	Cluster		Blockcluster		Transformation		$\mathcal{H}$-LU		vork. GMRes	
größe	#	t in s	#	t in s	t in s	MB	t in s	MB	#it	t in s
66 598	2 067	1.33	12 053	0.00	2.05	65	3.28	76	9	0.62
152 308	6 675	3.53	40 450	0.01	5.07	168	17.02	222	11	2.20
301 995	13 615	8.66	103 868	0.03	12.50	392	77.32	604	15	7.82
412 800	17 357	12.91	187 704	0.03	16.62	522	143.24	907	17	13.60
525 012	18 070	16.75	209 550	0.03	22.02	689	171.87	1 148	20	20.31
603 595	28 427	21.22	285 483	0.06	28.51	900	290.38	1 664	17	24.11
650 743	28 651	23.40	293 120	0.06	31.51	993	312.78	1 824	16	24.50
793 092	32 431	29.56	384 644	0.08	37.66	1 200	401.06	2 238	17	32.28
1 058 565	37 621	41.84	573 920	0.09	47.68	1 500	598.77	2 977	18	46.53

Tabelle 6.11: $\mathcal{H}$-LU-Zerlegung, $n_{\min} = 50$

vergleichbar große Testmatrizen relativ hoch. Der berechnete $\mathcal{H}$-LU-Vorkonditionierer ist aber sehr effizient, denn das GMRes(30)-Verfahren konvergiert nach wenigen Iterationen.

6.5 Parallele $\mathcal{H}$-Cholesky-Zerlegung

Um die Laufzeit der $\mathcal{H}$-LU-Zerlegung zu reduzieren, können einerseits schnellere Prozessoren eingesetzt werden. Andererseits kann die Arbeit auf mehrere Prozessoren verteilt werden. Bei der zweiten Variante unterscheidet man grob zwischen Systemen, bei denen die Prozessoren den Speicher gemeinsam nutzen (*shared memory*), und den Systemen, bei denen jeder Prozessor seinen eigenen Speicher besitzt (*distributed memory*).

Distributed-memory-Systeme können durch die Vernetzung von Standard-Personalcomputern aufgebaut werden. Zur Kommunikation zwischen den Rechnern hat sich MPI (message parsing interface) als Standardschnittstelle etabliert, die auch bei der parallelen $\mathcal{H}$-LU-Zerlegung zum Einsatz kommt.

Zur Bewertung paralleler Algorithmen nutzt man die Begriffe Beschleunigung und Effizienz. Sei t_p die Laufzeit eines parallelen Algorithmus auf p Prozessoren. Die Beschleunigung (engl. speedup) wird durch

$$S(p) = \frac{t_1}{t_p}$$

definiert, die Effizienz durch

$$E(p) = \frac{S(p)}{p}$$

beschrieben.

Um Aussagen zur Skalierung und zur Asymptotik zu erhalten, werden numerische Tests mit den Matrizen aus Tab. 5.1 des Beispiels 5.14 auf einem Cluster, bestehend aus zwei Rechnern mit jeweils acht Intel Xeon Prozessorkernen (3.2 GHz, 6 MB Cache pro Kern), durchgeführt.

Kriemann untersucht in seiner Dissertation [49] die Parallelisierung der $\mathcal{H}$-Matrix-Operationen. Die numerischen Resultate in [49] werden allerdings anhand der $\mathcal{H}$-LU-

Matrix-	$p=1$	$p=2$		$p=4$		$p=8$		$p=12$	
größe	t in [s]	t in [s]	E	t in [s]	E	t in [s]	E	t in [s]	E
101 296	2.67	1.47	0.91	0.84	0.79	0.51	0.65	0.54	0.41
297 927	10.02	5.39	0.93	2.90	0.86	1.64	0.76	1.74	0.48
658 609	23.00	12.84	0.90	7.19	0.80	3.93	0.73	4.25	0.45
906 882	39.73	20.84	0.95	11.20	0.89	10.33	0.48	6.56	0.50
2 539 954	122.05	63.03	0.97	34.27	0.89	18.74	0.81	19.85	0.51
5 413 520	283.36	151.22	0.94	79.04	0.90	42.50	0.83	45.52	0.52

Tabelle 6.12: parallele $\mathcal{H}$-Cholesky-Zerlegung, $\delta = 0.1$

Zerlegung für BEM-Matrizen gewonnen. Trotzdem ähneln der Speedup und die Skalierung der parallelen $\mathcal{H}$-Cholesky-Zerlegung den in Kapitel sechs der Dissertation angegebenen Resultaten. In der Tab. 6.12 sieht man die fallende parallele Effizienz für die zunehmende Anzahl von Prozessoren. Mit zunehmender Problemgröße steigt die parallele Effizienz leicht an.

6.6 Einsatz von AMG- und $\mathcal{H}$-Vorkonditionierern bei der Lösung von nichtlinearen FEM-Problemen

Bei der numerischen Lösung einer nichtlinearen partiellen Differentialgleichung entsteht eine nichtlineare Gleichung

$$F(x) = 0, \tag{6.1}$$

mit $F : D \subset \mathbb{R}^n \to \mathbb{R}^n$.

Sei F mindestens einmal stetig differenzierbar und die Jacobi-Matrix für $x \in D$ invertierbar. Dann führt die sukzessive angewandte Linearisierung von F (durch das Taylor-Polynom vom Grad 1) auf das *Newton-Verfahren*

$$F'(x_\nu)s_\nu = -F(x_\nu) \tag{6.2a}$$

$$x_{\nu+1} = x_\nu + s_\nu, \quad \nu = 0, 1, \ldots . \tag{6.2b}$$

Der Term s_ν wird als *Newton-Korrektur* bezeichnet. Die Lösung des nichtlinearen Problems (6.1) wird auf die Lösung einer Folge von linearen Problemen zurückgeführt. Unter der Voraussetzung, dass ein Startwert x_0 ausreichend nahe an der exakten Lösung x^* liegt, konvergiert das Newton-Verfahren quadratisch.

In jeder Iteration des Newton-Verfahrens wird die Jacobi-Matrix $F'(x_\nu)$ benötigt. Da die Assemblierung von $F'(x_\nu)$ meist aufwändig ist, wird diese in *Quasi-Newton-Verfahren* durch eine Approximation B_ν ersetzt:

$$B_\nu \bar{s}_\nu = -F(x_\nu)$$

$$x_{\nu+1} = x_\nu + \bar{s}_\nu, \quad \nu = 0, 1, \ldots .$$

Die Konvergenzgeschwindigkeit von Quasi-Newton-Verfahren ist superlinear, siehe [22].

Das *vereinfachte Newton-Verfahren* ist ein Quasi-Newton-Verfahren, in dem die initiale Jacobi-Matrix $F'(x_0)$ während des gesamten Newton-Verfahrens verwendet wird. Die

Einsparung an Kosten zur Berechnung von $F'(x_\nu)$, $\nu > 0$, führt aber zu einer geringeren Konvergenzgeschwindigkeit.

Die Lösung des linearen Systems (6.2a) mit direkten Methoden ist für großdimensionale Jacobi-Matrizen mit hohen Kosten verbunden. Mit iterativen Lösern berechnet man im Allgemeinen nicht die exakte Lösung, spart dadurch aber Rechenzeit pro Newton-Iteration:

$$\begin{aligned} F'(x_\nu)\tilde{s}_\nu &= -F(x_\nu) + r_\nu \\ x_{\nu+1} &= x_\nu + \tilde{s}_\nu, \quad \nu = 0, 1, \ldots . \end{aligned} \tag{6.3}$$

Dabei ist $r_\nu \in \mathbb{R}^n$ das Residuum der approximierten Lösung $\tilde{s_\nu}$. Da (6.2a) nur inexakt gelöst wird, werden die Verfahren als *inexakte Newton-Verfahren* bezeichnet. Die von nichtlinearen partiellen Differentialgleichungen stammenden linearen Probleme sind oft schlecht konditioniert. Daher muss die Konvergenz der iterativen Verfahren durch die Vorkonditionierung verbessert werden.

Zur Unterscheidung der Iterationen des Newton-Verfahrens und der Iterationen des linearen Lösers spricht man von äußeren und inneren Iterationen. Wird die Genauigkeit zur Lösung der linearen Systeme an die Genauigkeit der äußeren Iteration angepasst, so spricht man auch von *truncated Newton methods*. Die Anpassung sollte dabei das Konvergenzmuster des Newton-Verfahrens nicht wesentlich verändern, siehe [23, Abschnitt 1.3]. Unter der Voraussetzung

$$\frac{\|r_\nu\|}{\|F(x_\nu)\|} \leq \eta_\nu,$$

wobei die Folge $\{\eta_\nu\}$ uniform kleiner als Eins ist, sind die inexakten Newton-Verfahren superlinear konvergent, siehe [20]. Mit der Frage, wie genau die linearen Systeme (6.3) gelöst werden müssen, beschäftigt sich der Artikel [24].

Zur Konvergenz obiger Newton-Verfahren wird vorausgesetzt, dass sich die Startlösung x_0 ausreichend nahe an der Lösung x^* befindet. In der Praxis ist diese Voraussetzung nicht immer erfüllt. Mit dem Verfahren des steilsten Abstiegs oder Dämpfungsverfahren / Trust Region Methoden kann das Newton-Verfahren global konvergent gemacht werden. Kapitel drei von [23] behandelt verschiedene Globalisierungsstrategien für Newton-Verfahren.

Im Schwerpunktprojekt 1146 „Modellierung inkrementeller Umformverfahren“ der Deutschen Forschungsgemeinschaft werden die Prozesse in der Umformzone untersucht. Unter einem inkrementellen Umformverfahren versteht man ein Verfahren, bei dem ein Werkzeug wiederholt auf das Werkstück einwirkt. Die in der Umformzone auftretenden elasto-plastischen Prozesse werden durch nichtlineare partielle Differentialgleichungen beschrieben. Diese werden durch ein (inexaktes) Newton-Verfahren gelöst. Dabei ändern sich die Jacobi-Matrizen bis zur vollständigen Kontaktetablierung stark. Treten keine Änderungen des Kontakts zwischen Werkstück und Werkzeug auf, überwiegen die Nichtlinearitäten des Werkstoffs, d.h. die elasto-plastischen Eigenschaften. Die Jacobi-Matrizen ändern sich nicht mehr so stark.

Die Eigenschaften des vorliegenden Problems können zur effizienten Lösung ausgenutzt werden. Dazu wird im CG-Verfahren ein schnell berechenbarer Vorkonditionierer bei

sich stark ändernden Jacobi-Matrizen eingesetzt. Unterliegen die Jacobi-Matrizen keinen großen Änderungen mehr, dann kann ein qualitativ hochwertiger, wiederverwendbarer Vorkonditionierer für die inneren Iterationen des Newton-Verfahrens genutzt werden.

Auf Grund der geringen Setup-Zeit des algebraischen Mehrgitter-Verfahrens (AMG) setzen wir dieses in den ersten Newton-Iterationen als Vorkonditionierer ein. Im Gegensatz zum hierarchischen Vorkonditionierer ($\mathcal{H}$-Vorkonditionierer) kann der AMG-Vorkonditionierer jedoch nicht wieder verwendet werden. Dafür ist die Konstruktion der $\mathcal{H}$-LU-Zerlegung vergleichsweise aufwändig, aber die Lösung kann sehr schnell berechnet werden.

Das heisst, das vereinfachte Newton-Verfahren wird mit einem inexakten Newton-Verfahren kombiniert, siehe Alg. 21. Als Testumgebung dient das im SPP 1146 ver-

Algorithmus 21 linear Solve in Newton step n_{Newton}

```
1: if tangent matrix changed significantly then
2:     compute AMG preconditioner
3:     solve Newton step using CG with AMG as preconditioner
4: else
5:     if n_renew demands update then
6:         compute / renew H-LU decomposition
7:     solve Newton step using CG with H-LU decomposition as precond.
```

wendete FEM-Programm LARSTRAN/shape. Da für das eingesetzte AMG-Verfahren lediglich eine Lizenz für einen Pentium 4 (3 GHz) mit 2 GB Hauptspeicher zur Verfügung stand, wurden die Tests auf einem System durchgeführt, das nicht mehr dem Stand der aktuellen Rechentechnik entspricht.

Wir ersetzen den Aufruf des LARSTRAN-eigenen linearen Lösers innerhalb des Newton-Verfahrens durch den Alg. 21. Die in LARSTRAN voreingestellte Genauigkeit für das Newton-Verfahren beträgt 10^{-3}.

Zum Test des Algorithmus simulieren wir einen Reckschmiedeprozess. In der Abb. 6.2 ist eine Skizze des Modells dargestellt. Ein starres Werkzeug wirkt auf einen Stahlblock der Dimension $90 \times 20 \times 20$ ein. Das Werkzeug ist neun Einheiten breit und hat einen Kantenradius von drei. Der Vorschub in y-Richtung beträgt 2.5 Einheiten und wird in 20 Inkrementen über die Prozessdauer von 1 s aufgebracht.

Das Modell wird in zwei Diskretisierungsstufen genutzt. Die Größe der resultierenden FE-Matrizen ist 16 362 bzw. 65 912. Für asymptotische Aussagen wären weitere, feinere Diskretisierungen wünschenswert. Jedoch konnte aus technischen Gründen keine feinere Diskretisierung gerechnet werden. Die Abbildungen 6.3 und 6.4 geben die Rechenzeiten für verschiedene Parameterkombinationen wieder. Zum einen wurde der Umschaltzeitpunkt (n_{Newton}) von AMG-Vorkonditionierer auf $\mathcal{H}$-Vorkonditionierer variiert, d.h. ab dem Newton-Schritt n_{Newton} wird die $\mathcal{H}$-LU Zerlegung genutzt. Zum anderen wurden die Anzahl der Newton-Iterationen mit dem gleichen $\mathcal{H}$-Vorkonditionierer modifiziert (Achse n_{renew}). Wird im gesamten Newton-Verfahren das AMG-Verfahren zur Vorkonditionierung eingesetzt, so ist der Umschaltzeitpunkt ∞.

Um die Parameterkombination mit minimaler Rechenzeit besser darstellen zu können,

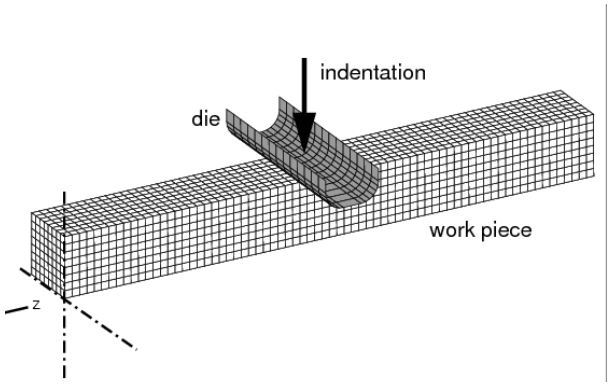


Abb. 6.2: Reckschmieden (kleine Diskretisierung)

sind die Beschriftungen der n_{Newton}-Achsen in der Abb. 6.3 gegenüber den Beschriftungen der Abb. 6.4 verändert.

Die geringsten Rechenzeiten für das kleine Modell erhält man mit den Parameterkombinationen $(n_{\text{Newton}}, n_{\text{renew}}) = (3, 15)$, $(n_{\text{Newton}}, n_{\text{renew}}) = (3, 10)$, $(n_{\text{Newton}}, n_{\text{renew}}) = (2, 10)$ und $(n_{\text{Newton}}, n_{\text{renew}}) = (2, 15)$. Der AMG-Vorkonditionierer wird bei diesen Parameterkombinationen für maximal zwei Newton-Schritte verwendet. Dann wird auf den $\mathcal{H}$-Vorkonditionierer umgeschaltet.

Für das große Modell erhält man die geringste Rechenzeit mit der Parameterkombination $(n_{\text{Newton}}, n_{\text{renew}}) = (1, 20)$, d.h. es wird lediglich der hierarchische Vorkonditionier eingesetzt. Aus den Abbildungen erkennt man, dass sich ein möglichst frühes Umschalten auf den hierarchischen Vorkonditionierer lohnt. Die zwei Beispiele belegen, dass hierarchische Vorkonditionierer in Kombination mit AMG-Vorkonditionierern im Newton-Verfahren effizient eingesetzt werden können.

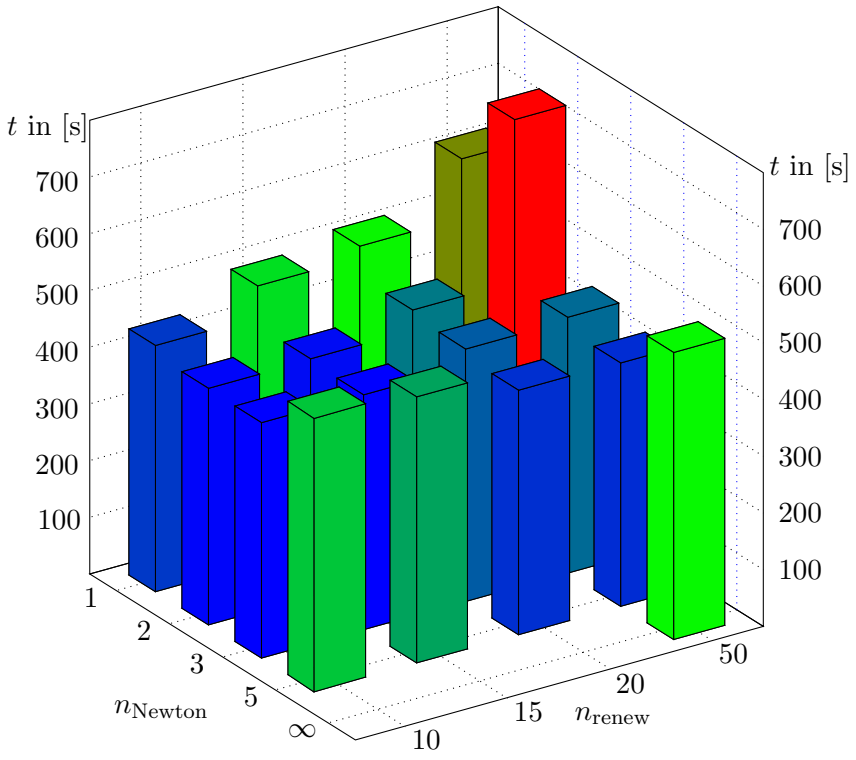


Abb. 6.3: Rechenzeiten Reckschmieden - kleines Modell

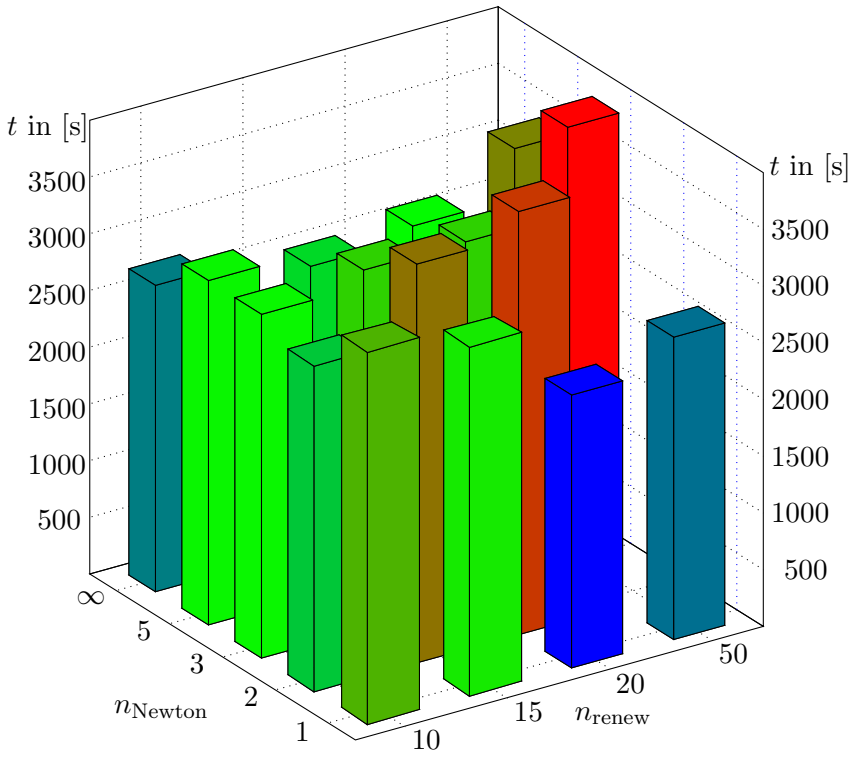


Abb. 6.4: Rechenzeiten Reckschmieden - großes Modell

7 Fazit und Ausblick

In dieser Arbeit wird eine approximative LU-Zerlegung zur schnellen Berechnung eines effizienten Vorkonditionierers für Finite-Elemente-Matrizen entwickelt und analysiert. Die Faktoren L und U werden im hierarchischen Format berechnet. Dazu wird die Matrix zunächst in Blöcke partitioniert und die Matrixeinträge entsprechend der Partition transformiert. Von der entstandenen $\mathcal{H}$-Matrix wird die $\mathcal{H}$-LU-Zerlegung berechnet. Dabei ist eine gute Partitionierung wichtig für die Laufzeit der arithmetischen Operationen.

Die theoretischen Ergebnisse in Kapitel 4 zeigen, dass der Vorkonditionierer in linear-logarithmischer Zeit generiert werden kann. Sowohl zum Beweis der Sätze als auch zur Konstruktion der Partition wurden bisher geometrische Informationen genutzt.

In dem im Kapitel 5 vorgestellten Verfahren wird nur die Matrix bzw. der Matrixgraph zur Konstruktion der $\mathcal{H}$-Matrix verwendet. Die Existenz der Faktoren L und U im hierarchischen Format konnte für gut konditionierte Matrizen bewiesen werden. Dazu wird eine algebraische Zulässigkeitsbedingung (5.4) eingeführt. Die numerischen Experimente zeigen, dass nicht nur gut konditionierte Probleme gelöst werden können.

Algorithmen zur effizienten Berechnung der Zulässigkeitsbedingung werden ebenfalls in Kapitel 5 angegeben. Es zeigt sich, dass die Algorithmen zur Abstandsbestimmung nur grobe Näherungen berechnen. Wie die numerischen Ergebnisse in Kapitel 6 zeigen, reichen diese Näherungen zur Konstruktion der $\mathcal{H}$-Matrizen und einem darauf basierenden effizienten Vorkonditionierers aus. Daher liegt die Vermutung nahe, dass die schwächere Zulässigkeitsbedingung (5.3) zur Berechnung eines Vorkonditionierers ausreicht.

Diese Lücke zwischen praktischem Verhalten und theoretischen Aussagen kann eventuell durch eine andere Herangehensweise zum Beweis der Darstellbarkeit der Faktoren L und U im hierarchischen Format geschlossen werden.

Trotz der für die Clusterung bzw. Block-Clusterung anspruchsvollen Geometrie aus Abb. 5.9 skalieren Zeit und benötigter Speicherplatz zur Berechnung der algebraischen Partition nahezu linear mit der Anzahl der Unbekannten.

Anhand des Beispiels in Abschnitt 6.1 haben wir gesehen, dass die algebraische Partitionierung gegenüber der geometrischen Partitionierung Vorteile sowohl im Speicherverbrauch als auch in der Rechenzeit hat. Dabei wirken die Vorkonditionierer etwa gleich effizient. Im Vergleich zu einem der effizientesten direkten Löser sowie einem AMG-Verfahren (Abschnitt 6.3) wird das asymptotische Verhalten der unterschiedlichen Löser deutlich. Die asymptotisch geringste Laufzeit des AMG-Verfahrens geht einher mit mangelnder Robustheit des Verfahrens. Zudem kann der AMG-Vorkonditionierer nur für eine rechte Seite eingesetzt werden. Im Gegensatz dazu erhöht sich die Robustheit des CG-Verfahrens mit hierarchischem Vorkonditionierer.

Methoden zur besseren Approximation der kleinen Eigenwerte werden derzeit in der Arbeitsgruppe Bebendorf an der Universität Bonn entwickelt. Die genauere Approxima-

tion der kleinen Eigenwerte stellt einen vielversprechenden Ansatz dar, um den Vorkonditionierer weiter zu verbessern.

Die Ergebnisse der Experimente zur Lösung nichtlinearer FEM-Probleme veranschaulichen, dass der Einsatz von $\mathcal{H}$-Vorkonditionierern im linearen Löser des Newton-Verfahrens zu geringen Rechenzeiten führt. Auf Grund der Wiederverwendbarkeit des $\mathcal{H}$-Vorkonditionierers ergeben sich im Vergleich zum ausschließlich AMG-vorkonditionierten linearen Löser im Newton-Verfahren kürzere Rechenzeiten. Die Experimente wurden im FEM-Programm LARSTRAN/shape auf einem Pentium 4 Prozessor (3 GHz) durchgeführt. Mit aktueller Rechentechnik wäre der Test feinerer Diskretisierungen und eine breitere Variation der Parameter möglich.

Literaturverzeichnis

[1] P. R. Amestoy, T. A. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM J. Matrix Anal. Appl.*, 17(4):886–905, 1996.

[2] P. R. Amestoy, T. A. Davis, and I. S. Duff. Algorithm 837: AMD, an approximate minimum degree ordering algorithm. *ACM Trans. Math. Software*, 30(3):381–388, 2004.

[3] O. Axelsson and P. Vassilevski. Algebraic multilevel preconditioning methods I. *Numer. Math.*, 56:157–177, 1989.

[4] O. Axelsson and P. Vassilevski. Algebraic multilevel preconditioning methods II. *SIAM J. Numer. Anal.*, 27:1569–1590, 1990.

[5] N. Bakhvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Computational Math. and Math. Phys*, 6:101–135, 1966.

[6] M. Bebendorf. Efficient inversion of Galerkin matrices of general second-order elliptic differential operators with nonsmooth coefficients. *Math. Comp.*, 74:1179–1199, 2005.

[7] M. Bebendorf. Approximate Inverse Preconditioning of FE Systems for Elliptic Operators with Non-smooth Coefficients. *SIAM J. Matrix Anal. Appl.*, 27(4):909–929, 2006.

[8] M. Bebendorf. Why finite element discretizations can be factored by triangular hierarchical matrices. *SIAM J. Num. Anal.*, 45(4):1472–1494, 2007.

[9] M. Bebendorf. *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, volume 63 of *Lecture Notes in Computational Science and Engineering (LNCSE)*. Springer-Verlag, 2008. ISBN 978-3-540-77146-3.

[10] M. Bebendorf and T. Fischer. On the Purely Algebraic Data-Sparse Approximation of the Inverse and the Triangular Factors of Sparse Matrices. 2010. to appear in NLA.

[11] M. Bebendorf and W. Hackbusch. Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients. *Numer. Math.*, 95(1):1–28, 2003.

[12] M. Bebendorf and W. Hackbusch. Stabilized rounded addition of hierarchical matrices. *Numer. Lin. Alg. Appl.*, 14(5):407–423, 2007.

[13] M. Bebendorf and R. Kriemann. Fast parallel solution of boundary integral equations and related problems. *Comput. Visual. Sci.*, 8:121–135, 2005.

[14] M. Benzi. Preconditioning techniques for large linear systems: A survey. *Journal of Computational Physics*, 182:418–477, 2002.

[15] M. Benzi and M. Tůma. A comparative study of sparse approximate inverse preconditioners. *Applied Numerical Mathematics*, 30:305–340, 1999.

[16] D. Braess. *Finite Elemente - Theorie, schnelle Löser und Anwendungen in der Elastizitätstheorie*. Springer, Berlin Heidelberg New York, 3 edition, 2002.

[17] A. Brandt. Multi-level adaptive solution to boundary-value problems. *Math. Comp.*, 31:333–390, 1977.

[18] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Electrical Engineering and Computer Science. MIT Press, 1998.

[19] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, pages 157–172, New York, NY, USA, 1969. ACM.

[20] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. Inexact newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[21] J. W. Demmel, N. J. Higham, and R. S. Schreiber. Stability of block LU factorization. *Numerical Linear Algebra with Applications*, 2:85 – 194, 1995.

[22] J. E. Dennis and J. J. Moré. A characterization of superlinear convergence and its applications to quasi-Newton methods. *Math. Comp.*, 28:549–560, 1974.

[23] P. Deuflhard. *Newton methods for nonlinear problems: affine invariance and adaptive algorithms*. Number 35 in Springer series in computational mathematics. Springer, 2004. ISBN-10: 3540210997 ISBN-13: 978-3540210993.

[24] S. C. Eisenstat and H. F. Walker. Choosing the Forcing Terms in an Inexact Newton Method. *SIAM Journal on Scientific Computing*, 17(1):16–32, 1996.

[25] R. Fedorenko. A relaxation method for solving elliptic difference equations. *USSR Computational Math. and Math. Phys.*, 1:1092–1096, 1962.

[26] R. Fedorenko. The speed of convergence of an iterative process. *USSR Computational Math. and Math. Phys.*, 4:227–235, 1964.

[27] M. Fiedler. Algebraic connectivity of graphs. *Czech. Math. J.*, 23:298–305, 1973.

[28] M. Fiedler. A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czech. Math. J.*, 25:619–633, 1975.

[29] A. George. *Computer implementation of the finite element method.* PhD thesis, Stanford, CA, USA, 1971.

[30] A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10(2):345–363, 1973.

[31] A. George and J. Liu. The evolution of the Minimum Degree Ordering Algorithm. *SIAM Review*, 31:1–19, 1989.

[32] G. H. Golub, C. F. V. Loan, and Golub. *Matrix computations.* Johns Hopkins series in the mathematical sciences. Johns Hopkins Univ. Pr., Baltimore, Md. [u.a.], 3 edition, 1996. ISBN-10: 0801854148 ISBN-13: 978-0801854149.

[33] L. Grasedyck. *Theorie und Anwendungen Hierarchischer Matrizen.* PhD thesis, Universität Kiel, 2001.

[34] L. Grasedyck, R. Kriemann, and S. L. Borne. Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems. *Computing and Visualization in Science*, 11(4-6):273–291, September 2008.

[35] W. Hackbusch. On the multi-grid method applied to difference equations. *Computing*, 20(4):291–306, 12 1978.

[36] W. Hackbusch. *Iterative Lösung großer schwachbesetzter Gleichungssysteme.* Teubner Studienbücher: Mathematik. Teubner Verlag, Stuttgart, 1993.

[37] W. Hackbusch. *Theorie und Numerik elliptischer Differentialgleichungen.* Teubner Verlag, 2 edition, 03 1997.

[38] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[39] W. Hackbusch. *Hierarchische Matrizen: Algorithmen und Analysis.* Springer Berlin Heidelberg, 2009.

[40] W. Hackbusch and B. N. Khoromskij. A sparse $\mathcal{H}$-matrix arithmetic: general complexity estimates. *J. Comput. and Appl. Math.*, 125(1-2):479–501, 2000.

[41] W. Hackbusch and B. N. Khoromskij. A sparse $\mathcal{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems. *Computing*, 64(1):21–47, 2000.

[42] W. Hackbusch, B. N. Khoromskij, and R. Kriemann. Hierarchical matrices based on a weak admissibility criterion. *Computing*, 73:207–243, 2004.

[43] B. Hendrickson and E. Rothberg. Improving the run time and quality of nested dissection ordering. *SIAM J. Sci. Comput.*, 20(2):468–489 (electronic), 1998.

[44] V. E. Henson and U. M. Yang. BoomerAMG: a parallel algebraic multigrid solver and preconditioner. *Appl. Numer. Math.*, 41(1):155–177, 2002.

[45] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[46] I. Ibragimov, S. Rjasanow, and K. Straube. Hierarchical Cholesky decomposition of sparse matrices arising from curl-curl-equation. *Journal of Numerical Mathematics*, 15(1):31–57, 2007.

[47] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1999.

[48] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *The Bell System Technical Journal*, 29, 1970.

[49] R. Kriemann. *Parallele Algorithmen für $\mathcal{H}$-Matrizen.* PhD thesis, Christian-Albrechts-Universität zu Kiel, 2004.

[50] R. Kriemann. Parallel $\mathcal{H}$-matrix arithmetics on shared memory systems. *Computing*, 74(3):273–297, 2005.

[51] G. Meinardus. *Approximation of Functions: Theory and Numerical Methods.* Springer New York, 1967.

[52] G. L. Miller, S.-H. Teng, W. Thurston, and S. A. Vavasis. Geometric separators for finite-element meshes. *SIAM J. Sci. Comput.*, 19(2):364–386 (electronic), 1998.

[53] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quarterly Journal of Mathematics*, 11(1):50–59, 1960.

[54] J. A. S. Nicholas I. M. Gould and Y. Hu. A numerical evaluation of sparse direct solvers for the solution of large, sparse, symmetric linear systems of equations. *ACM Transactions on Mathematical Software*, 33 (2), 2007.

[55] A. Pothen, H. D. Simon, and L. Wang. Spectral Nested Dissection. Technical Report RNR-92-003, Nasa Ames Research Center, Moffett Field, CA 94035, 1992.

[56] D. J. Rose. A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations. In R. C. Read, editor, *Graph Theory and Computing*, pages 183–217. Academic Press, 1972.

[57] J. W. Ruge and K. Stüben. *Algebraic multigrid*, volume 3 of *Frontiers in applied mathematics*, chapter 4, pages 73–130. SIAM, Philadelphia, 1987.

[58] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, 2 edition, 1996. ISBN-10: 0898715342 ISBN-13: 978-0898715347.

[59] O. Schenk and K. Gärtner. Solving unsymmetric sparse systems of linear equations with PARDISO. *Future Gener. Comput. Syst.*, 20(3):475–487, 2004.

[60] O. Schenk and K. Gärtner. On fast factorization pivoting methods for symmetric indefinite system. *Elec. Trans. Numer. Anal.*, 23:158–179, 2006.

[61] O. Steinbach. *Lösungsverfahren für lineare Gleichungssysteme*. Mathematik für Ingenieure und Naturwissenschaftler. Vieweg+Teubner Verlag, 2005.

[62] G. Windisch. *M-matrices in numerical analysis*, volume 115 of *Teubner-Texte zur Mathematik*. Teubner, Leipzig, 1989.

[63] M. Yannakakis. Computing the minimum fill-in is NP-complete. *SIAM J. Algebraic Discrete Methods*, 2:77–79, 1981.

Selbstständigkeitserklärung

Hiermit erkläre ich, die vorliegende Dissertation selbstständig und ohne unzulässige fremde Hilfe angefertigt zu haben. Ich habe keine anderen als die angeführten Quellen und Hilfsmittel benutzt und sämtliche Textstellen, die wörtlich oder sinngemäß aus veröffentlichten oder unveröffentlichten Schriften entnommen wurden, und alle Angaben, die auf mündlichen Auskünften beruhen, als solche kenntlich gemacht. Ebenfalls sind alle von anderen Personen bereitgestellten Materialien oder erbrachten Dienstleistungen als solche gekennzeichnet.

Leipzig, 26. Mai 2010

Wissenschaftlicher Werdegang

12/2005	–	11/2009	**Promotionsstudent / wissenschaftlicher Mitarbeiter** im Fachbereich Numerik der Universität Leipzig Forschungsgebiet: iterative Lösung großer schwachbesetzter linearer Gleichungssysteme, Implementierung in C++
07/2004	–	11/2005	**Wissenschaftlicher Mitarbeiter** der FH Erfurt Aufbau eines elearning Portals Betreuung von Übungen am Computer
09/1997	–	06/2003	**Studium der Informatik** an der FSU Jena Vertiefung: technische und angewandte Informatik Abschluss: Diplom-Informatiker mit Gesamtprädikat sehr gut

Vorträge und Posterpräsentationen

02/2007	Vortrag beim Zwischenkolloquium des DFG SPP 1146 zum Thema *Neue Clusterstrategie zur effizienten Berechnung von Vorkonditionierern mit hierarchischen Matrizen*
08/2007	Vortrag beim Arbeitskreistreffen des DFG SPP 1146 zum Thema *Vorkonditionierung mit hierarchischen Matrizen*
09/2007	Vortrag auf der ENUMATH2007, Graz zum Thema *An algebraic approach to preconditioning based on $\mathcal{H}$-LU decompositions*
02/2008	Posterpräsentation beim Zwischenkolloquium des DFG SPP 1146 zum Thema *An algebraic approach to the efficient computation of preconditioners using Hierarchical Matrices*
04/2008	Vortrag beim Südostdeutschen Kolloquium zum Thema *Algebraische Vorkonditionierung mit Hierarchischen Matrizen*
04/2009	Vortrag beim Arbeitskreistreffen des DFG SPP 1146 zum Thema *Vorkonditionierung mit Hierarchischen Matrizen*

Veröffentlichungen

T. Fischer. Simulation des Abtragsverhaltens eines realen Polierwerkzeugs. Master's thesis, Friedrich-Schiller-Universität Jena, 05 2003.

L. Ihring and T. Fischer. www.bildung-bau.de - Konzept einer Lernplattform mit Community-Charakter und die technische Unterstützung durch das LMS moodle. In *Proceedings of the Workshop on e-Learning 2005: Auf dem Weg vom Hype in die Konsolidierung?*, pages 43–56. HTWK Leipzig, Volker Dötsch, Klaus Hering, Florian Schaar, Juli 2005.

M. Bebendorf and T. Fischer. A Purely Algebraic Approach to Preconditioning based on Hierarchical LU Factorizations. In *Numerical Mathematics and Advanced Applications*, pages 135–142. Springer-Verlag, 2008.

M. Bebendorf and T. Fischer. On the Purely Algebraic Data-Sparse Approximation of the Inverse and the Triangular Factors of Sparse Matrices. 2010. to appear in NLA.